Doctoral Dissertation

Towards Fast and High-quality Biomedical Image Reconstruction

Tran Minh Quan

School of Electrical and Computer Engineering

The Graduate School of UNIST

2019

Towards Fast and High-quality Biomedical Image Reconstruction

Tran Minh Quan

School of Electrical and Computer Engineering

Graduate School of UNIST

Towards Fast and High-quality Biomedical Image Reconstruction

A dissertation
submitted to the the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Tran Minh Quan

12/21/2018

Approved by

Advisor
Won-Ki Jeong

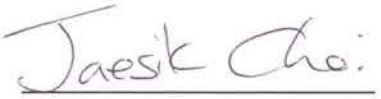
Towards Fast and High-quality Biomedical Image Reconstruction

Tran Minh Quan

This certifies that the dissertation of Tran Minh Quan is approved.

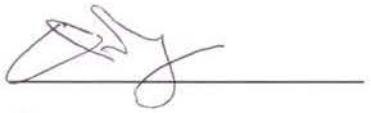
12/21/2018

Won-Ki Jeong (UNIST): Advisor

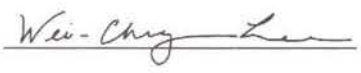


Jaesik Choi (UNIST): Thesis Committee Member #1

Se Young Chun (UNIST): Thesis Committee Member #2



Hyung Joon Cho (UNIST): Thesis Committee Member #3



Wei-Chung Allen Lee (HMS): Thesis Committee Member #4

Abstract

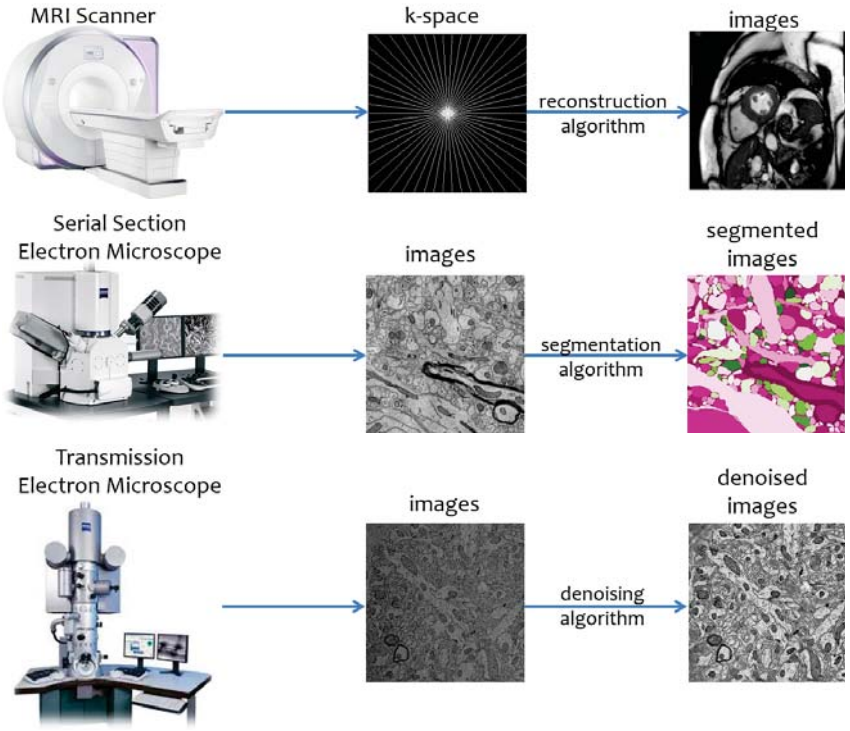


Figure 1: Image reconstruction tasks can be varied across modalities.

Reconstruction is an important module in the image analysis pipeline with purposes of isolating the majority of meaningful information that hidden inside the acquired data. The term "reconstruction" can be understood and subdivided in several specific tasks in different modalities. For example, in bio-medical imaging, such as Computed Tomography (CT), Magnetic Resonance Image (MRI), that term stands for the transformation from the, possibly fully or under-sampled, spectral domains (sinogram for CT and k-space for MRI) to the visible image domains. Or, in connectomics, people usually refer it to segmentation (reconstructing the semantic contact between neuronal connections) or denoising (reconstructing the clean image). Figure 1 is a pictorial description of the above image reconstruction applications.

In this dissertation research, I will describe a set of my contributed algorithms from conventional to state-of-the-art deep learning methods, with a transition at the data-driven dictionary learning approaches that tackle the reconstruction problems in various image analysis tasks.

Contents

List of Figures

List of Tables

I Introduction

1.1 Motivation

Mankind has always been curious and eager to understand everything around it. We start with rudimentary concepts and expand it in order to answer the questions about our unknown universe. With that persevering ambition, we have also developed essential tools to follows the trends. This argument is commonly-known as **"Technologies push, applications pull"** which have motivated us to explore our dynamic environment whose data (or its representation) keeps increasing vastly as time goes by. However, our senses are partially limited in their abilities, especially the vision system. For example, our human eyes cannot perceive waves that are outside of the visible lights spectrum such as radio, microwave, infrared, or ultraviolet, X-ray, gamma, etc. However, by mapping the observations, or **measurements**, such things can be quantitatively evaluated, e.g., energy range or frequencies of wavelength, we know that those invisibilities, or **insight**, exist. At this moment, it is worth defining two separated domains: observations (**measurements**) and knowledge (**insight**) in which the former can be collected via developing tools and the latter should be explainable to improve the understanding. Transforming inbetween these two spaces can be both relatively simple or complex. This task depends on how the measurements are acquired and how much for the insight should elevate up to. For instances, with an ambitious goal of seeing things indoor or outdoor, it is straightforward to use an identity matrix to map the natural scene(s) via a shutter (or eyes) to form photo(s) that stored on a film by burning the phosphor layers or image(s) by sending signals through our vision rod and cone cells to our brains. A little more challenging, with another ambitious goal of seeing things inside our bodies, we can collect the measurements via CT/MRI scanners, and map those signals (k-space) to the ones observable (visible image space) with a filter back projection/an inverse Fourier transform. On an abstract aspect, with the ambitious goal of observing clean things, we can define noisy and noise-free domains and the transformations inbetween are known as noise addition/removal tasks. Even more abstract, with the ambitious goal of observing distinct things (semantic-aware or instance-aware), we need to find proper segmentation algorithms to perform on the observations (images, users behaviors, etc.). At this time, the transformations are no-longer linear and complicated to design. They are also computationally-expensive since many regularizers (constraints) of our prior-knowledge need to be satisfied. Therefore, finding such those insights systematically as well as algorithmically based on some domain-specific observations poses some challenges and motivates me to work on this dissertation.

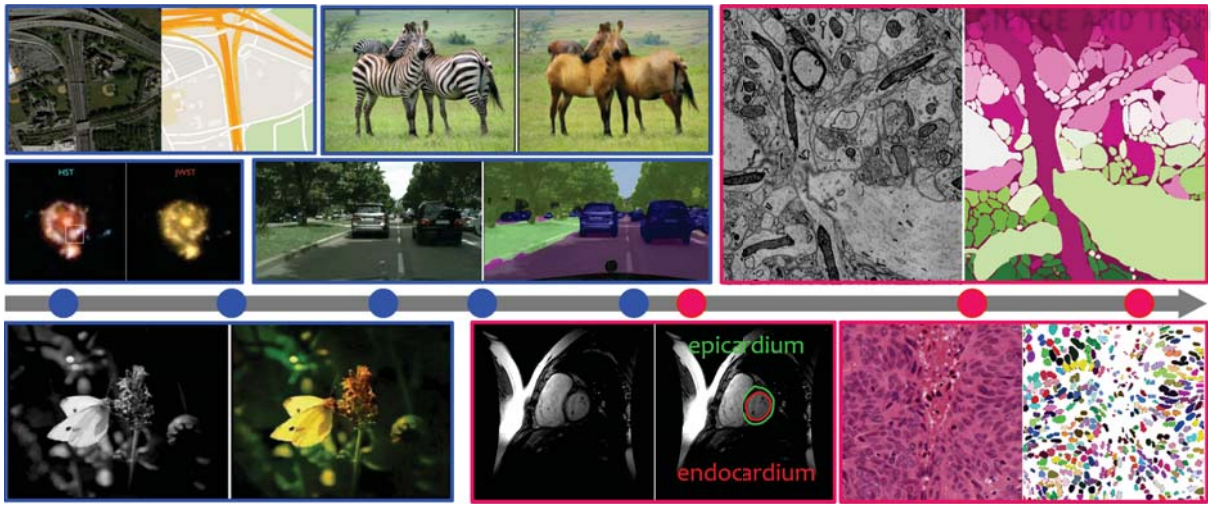


Figure 2: Image-to-image translation tasks can be varied across scales.

1.2 Problem statement

By narrowing down the **observations** and the **insights** to discrete and array-like representations, we can form an overview of image-to-image translation tasks (Figure 2). In a macro aspect, some examples can be referred as translating the optically astronomical observations to find the new planets or stars, or translating the satellite terrain image to traversal map. In a common aspect, making visual photos more plausible (image super-resolution, converting grayscale to color images, etc.) or helping to improve the autonomous driving systems are worth mentioning. In a micro aspect, especially in biomedical and clinical research, segmenting the object of interests (left/right ventricles, cell nuclei, neuronal structures) is also belonging to image-to-image translation. In the above examples, biomedical image data draw much attention due to its wide range of modalities, scales, and many more. And the mapping of such **measurements** from laboratories equipment to understand the image data is now interchangeable with **reconstruction**. By framing the image reconstruction into image-to-image translation framework, it is crucial to define **two or more image domains** that hold the source of **measurements** (e.g., k-space data, raw EM images, etc.) and the target **insights** (e.g., MR images, neuronal segmentation, clean EM images, etc.) with/without prior **constraints**. Furthermore, as the original problem is highly ill-posed, such translations need not to be unique. It means that some **insights** may have a lot of **measurements**. For example, the full MR images may have various kinds of under-sampled k-space appearances, or a skeleton of one single neural structure may have various types of textures on it as time goes by, or some clean images may have different types of noise-altered sources. Therefore, translations in image reconstruction is highly unbalanced as mapping inbetween domains can be all-to-one and vice versa. Among the possible paths, there should be ones that can inversely reveal the **insights**. These trajectories might be iterative, i.e.,

loosely refining the reconstructions so that the original **measurements** are always satisfying; or one-time predictive phase by leveraging the power of universal approximators (also known as neural networks) to shift the time-consuming part onto the training phase. Hence, the main goals of this dissertation are to develop faster and high-quality biomedical image reconstruction methods, which are representing as three tasks: restoring the image signals from undersampled Magnetic Resonance k-space data, segmenting the semantic information (membrane, blob-like cell nuclei), and removing noises from electron microscope images.

1.3 Example of image reconstruction problem: Compressed Sensing MRI

Magnetic resonance imaging (MRI) has been widely used as an in-vivo imaging technique because it is a non-intrusive high-resolution imaging technology safe to living organisms. Even though MRI does not use dangerous radiation for imaging, it usually requires a long acquisition time for high-resolution image reconstruction. This is a significant drawback of MRI in a clinical setup because a longer acquisition time causes discomfort to patients, and in some time-critical situations, such as emergency diagnosis, it may not be usable at all. There have been many approaches to reduce the acquisition time or increase the image resolution by using parallel acquisition hardware [10]. Recently, the Compressed Sensing (CS) theory [37] has been successfully adopted to MRI reconstruction [86] to reduce the amount of data needed to acquire while maintaining the image quality close to the full reconstruction. The idea of CS reconstruction is that if the original signal is sparse (or at least we can make it sparse by applying some transformations), the full-resolution signal can be computationally reconstructed from an under-sampled signal at a sampling rate below its Nyquist rate. Therefore, in order to apply the CS theory to MRI reconstruction, we need to find a proper sparse transformation to make the signal sparse, e.g., wavelet transformation, and solve a ℓ_1 minimization problem.

In this section, we will briefly overview the basics of Compressed Sensing MRI. We use the subscript f to denote the result of applying a masked Fourier transform to a given variable. We denote the under-sampled raw MRI data (k-space measurements) using the sampling mask R as m. Then, its zero-filling reconstruction s_0 can be obtained by the following equation:

$$s_0 = F^H R^H (m) \tag{1}$$

where F is the Fourier operator, and superscript H indicates the conjugated transpose of a given operator. Similarly, turning any image s_i into its under-sampled measurement m_{s_i} with the given sampling mask R can be done via the inverse of the reconstruction process:

$$m_{s_i} = RF (s_i) \tag{2}$$

Then, compressed sensing MRI reconstruction, which is a process of generating a full-reconstruction image s from under-sampled k-space data m, can be described as follows:

$$\min_{s} J(s) \quad s.t. \quad RF(s) = m \tag{3}$$

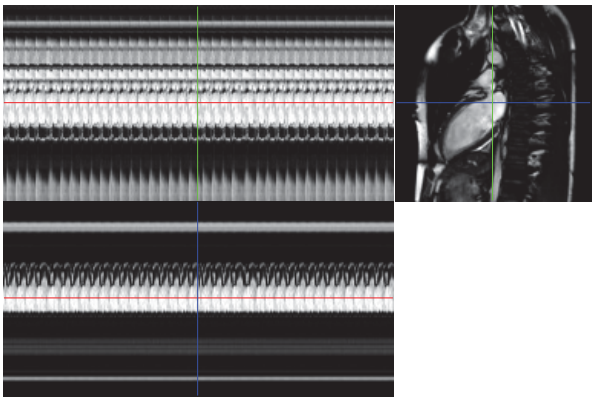
where F is the Fourier operator, R is the sampling mask, and superscript H indicates the conjugated transpose of a given operator. and the above constrained problem can be reformulated in an unconstrained fashion with weighting parameters as follows:

$$\min_{s} \|RF(s) - m\|_2^2 + \lambda J(s) \tag{4}$$

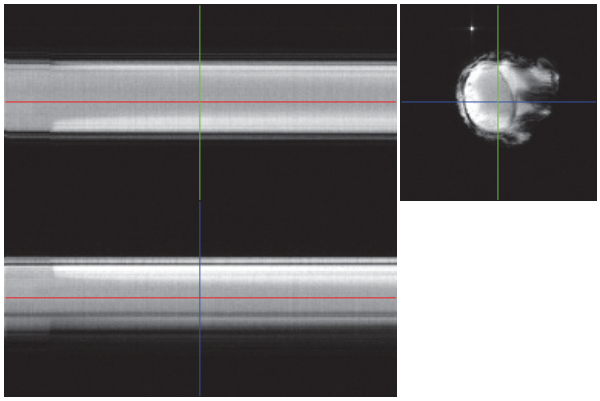
where $J(s)$ is a regularizer required for ill-posed optimization problems.

Many classical priors can be used for $J(s)$, such as Tikhonov regularization (IID Gaussian prior) [25], edge-preserving regularization [120], total variation (TV) [27], non-local mean filter (NLM) [90], wavelets [98], curvelets [8], etc. By enforcing ℓ_p norm where $0 \leq p \leq 1$ for regularization, compressed sensing theory [21, 37] can be applied to MRI reconstruction from highly undersampled k-space data [87]. This sparsity-induced image reconstruction method aims to find the solution (images s) that satisfies not only the under-sampled measurement constraints but also sparsity in the transformed domain by decomposing the signals with the designated universal transform sparsity basis. For example, the seminal work by Lustig *et al.* [86] proposed that $J(s)$ is equal to $\|\Psi s\|_0$, in which Ψ is the wavelet transform, and further relaxed the ℓ_0 norm by ℓ_1 norm substitution. Solving such nonlinear optimization problems usually involves an iterative process to minimize the data consistency energy and the sparsity energy. The long acquisition time is a fundamental challenge in MRI, so the compressed sensing (CS) theory has been proposed and successfully applied to speed up the acquisition process. Conventional CS-MRI reconstruction methods have been developed to leverage the sparsity of signal by using universal sparsifying transforms, such as Fourier transform, Total Variation (TV), and Wavelets [87], and to exploit the spatio-temporal correlations, such as $k-t$ FOCUSS [70, 71] This sparsity-based CS-MRI method introduces computational overhead in the reconstruction process due to solving expensive nonlinear ℓ_1 minimization problem, which leads to developing efficient numerical algorithms [51] and adopting parallel computing hardware to accelerate the computational time [99]. The nuclear norm and low-rank matrix completion techniques have been employed for CS-MRI reconstruction as well [95, 123, 135] .

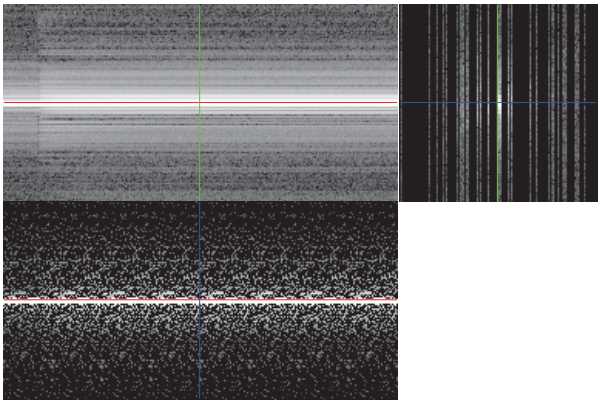
In addition, MRI can be used to monitor dynamic changes of the subject (Figure 3). Such dynamic MRI can be classified into two groups. One is that the structure of the subject is moving, as in MRI with a cardiac motion (Figure 3 (a)). In this data, there is a periodic

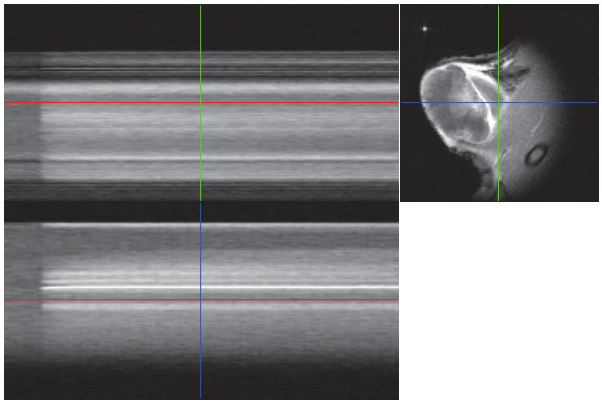


(a) Cardiac MRI



(b) DCE MRI





(c) CS-MRI reconstruction (right) from ×4 subsampled k-space data (left)

Figure 3: MRI examples. Red line : time axis, Blue-green lines : x-y axis, (a) and (b) : different types of time-varying dynamic MRI, (c) CS-MRI example

ripple in the temporal profile (along the red axis in the figure). The other is that the structure is not deforming but the pixel brightness is changing due to the reaction of the subject with injected chemicals (Figure 3 (b)). Such data is called Dynamic Contrast Enhanced (DCE) MRI. In both cases, high temporal resolution is important to resolve the subtle changes over time in dynamic MRI, and therefore the acquisition of a large amount of data is necessary. Even though CS reconstruction can accelerate the MRI acquisition process by reducing the data size, it introduces a computational overhead because ℓ_1 minimization is a time-consuming nonlinear optimization problem. Moreover, the reconstruction algorithm iteratively applies Fourier and wavelet transforms, which is also a very time-consuming process. Therefore, there exists a need to develop fast CS reconstruction methods to make the entire MRI reconstruction process practical for time-critical applications.

1.4 Inspirations

Over the past decade, the graphics processing unit (GPU) has evolved into a powerful computing platform for high-performance computation. It has opened a new direction for visual computing so that programmers can potentially increase their applications in both programmability and performance. On the other hand, science and engineering researchers across many disciplines have access to exponentially increasing amounts of computing power and research data available to support their work. State of the art high-performance resources are needed to tackle large computational problems and to make massive data accessible for superior analysis. Date back to 2012, bio-medical imaging has been one of the most premier fields which adopts high-performance accelerators such as GPUs to speed up its applications. Therefore, I plans to leverage the high-performance and massive computing platform such as a GPU cluster for implementing several image reconstruction algorithms in diverse acquisition contexts.

1.5 Research statement

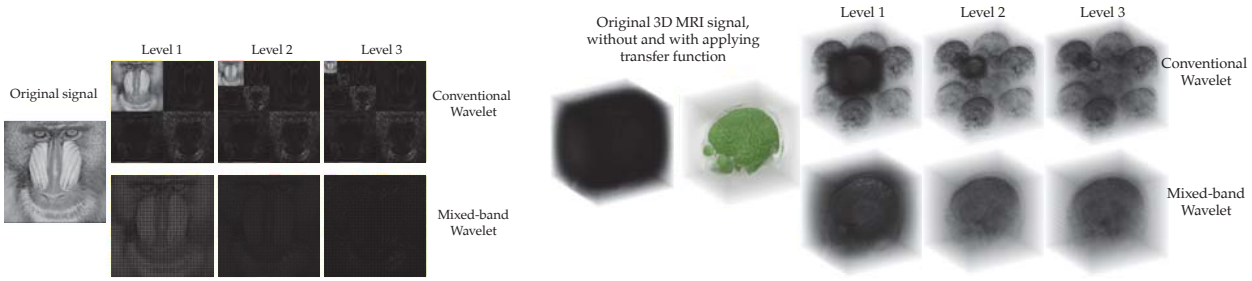
This work is centered around GPU-accelerated computing on biomedical image processing such as classical smoothing methods for undersampled MRI data using sparsifying transforms such as Wavelet transform, or recently leveraging data-driven machine learning methods to deliver a higher quality of the reconstructed MRI. My developed algorithms focused on unsupervised learning techniques by introducing a fast alternating method for reconstructing highly undersampled dynamic MRI data using convolutional sparse coding which can be 2D or 3D. The proposed solution leverages Fourier Convolution Theorem to accelerate the process of learning a set of filters and iteratively refine the MRI reconstruction based on the sparse codes found subsequently. In contrast to conventional Compressed Sensing methods which exploit the sparsity by applying universal transforms such as wavelet and total variation, my approach extracts and adapts the related information directly from the MRI data using compact shift-invariant filters. The reconstruction outperforms CPU implementation of the state-of-the-art dictionary learning-based approaches by up to two orders of magnitude. Recently, I have been working on Deep learning-based Generative Adversarial Networks (GANs) for reconstructing Compressed Sensing MRI and achieved the results that outperformed others in term of quality as well as running times.

Beside the primary work on MRI modality, I also involved in part of the connectomics data processing such as neuronal segmentation and denoising electron microscope data. The proposed method leverages the latest advances in machine learning, such as semantic segmentation and

residual neural networks, with a novel introduction of summation-based skip connections to allow a much deeper network architecture for a more accurate segmentation, either membrane- or blob-type delineation. Along side with adversarial training and the introduction of noise-patch concatenation, an improvement over the unpaired image-to-image translation using cycle-consistent loss is presented to show that having the prior-knowledge on the noise pattern such as gridtape film or charge damage is advantageous for the denoising process, compared to direct translation back and forth inbetween noisy and noise-free image domains.

II Background and Related Work

2.1 Discrete Wavelet Transform



(a) 2D image (b) 3D volume

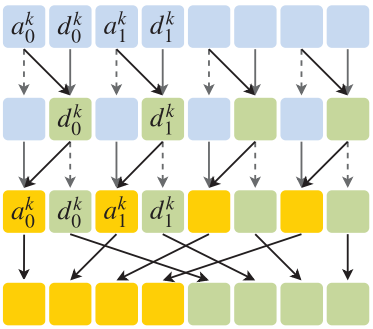
Figure 4: Examples of conventional (top row) and mixed-band (bottom row) wavelet transform.

The discrete wavelet transform (DWT) has been actively studied in the image processing domain and is also commonly used as a sparsifying transform in the Compressed Sensing reconstruction theory [87] in order to reduce noise or outliers in the reconstructed signal. One example is image compression: One can discard less-important information from a wavelet-transformed image more effectively, as in JPEG2000 [112]. The Federal Bureau of Investigation (FBI) uses a wavelet-based compression method for their fingerprint image database, which is one of the most effective biometric authentication techniques [13]. Another example is feature detection: One can develop an edge-detection filter via leveraging the high-pass filter of a wavelet transform [89]. Figure 4 illustrates the results of applying DWT to a typical 2D image and 3D medical data. More comprehensive reviews of wavelet transforms and their applications can be found in [88] and [35].

Not only has DWT been widely used in various disciplines, but also many of wavelet's applications are time-critical, such as in processing streaming video or the real-time reconstruction of sensor data. In order to accelerate wavelet transforms for such those purposes, special hardware or accelerators have been used. These include Field Programmable Gate Arrays (FPGAs), [36], [64], Intel Many Integrated Core (MIC) architecture, [7], and Graphics Processing Units (GPUs). Among them, GPUs have gained much attention due to their superior performance in terms of cost and energy consumption. High-level GPU programming APIs, such as NVIDIA CUDA [1] and OpenCL [2], have also promoted wide adoption of GPUs by lowering the learning curve.

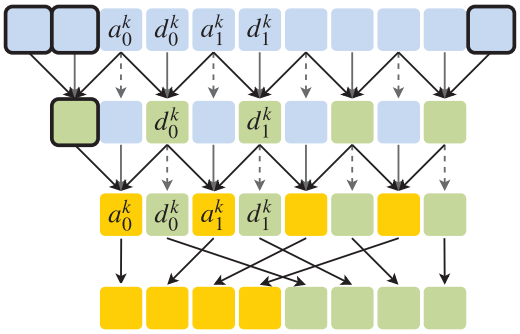
[1] `http://docs.nvidia.com/cuda/cuda-c-programming-guide/`

[2] `http://www.khronos.org/opencl/`



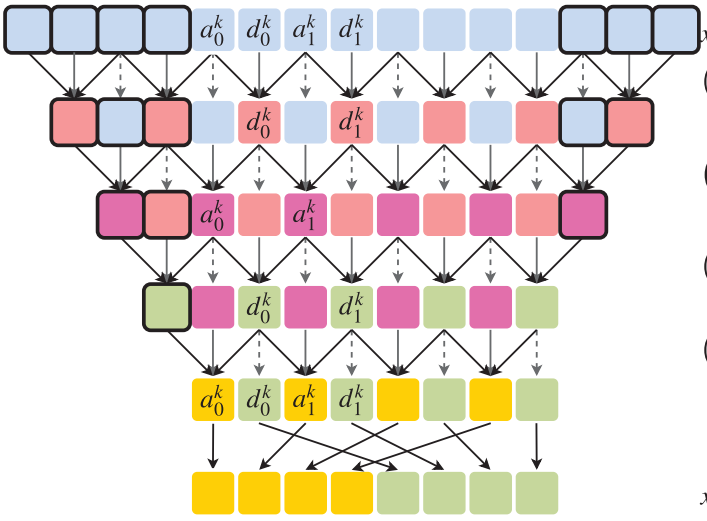
$$x^{k-1}[n] = \left\{\cdots x_j^{k-1} \cdots\right\}$$
$$\left(L_P^{\alpha}\right) : d_j^k = d_j^k - \alpha\left(a_j^k + a_{j+1}^k\right)$$
$$\left(L_U^{\beta}\right) : a_j^k = a_j^k + \beta\left(d_j^k + d_{j+1}^k\right)$$
$$x^k[n] = \left\{\cdots x_j^k \cdots\right\}$$

(a) Haar wavelet



$$x^{k-1}[n] = \left\{\cdots x_j^{k-1} \cdots\right\}$$
$$\left(L_P^{\alpha}\right) : d_j^k = d_j^k - \alpha\left(a_j^k + a_{j+1}^k\right)$$
$$\left(L_U^{\beta}\right) : a_j^k = a_j^k + \beta\left(d_j^k + d_{j+1}^k\right)$$
$$x^k[n] = \left\{\cdots x_j^k \cdots\right\}$$

(b) CDF 5/3 wavelet



$$x^{k-1}[n] = \left\{\cdots x_j^{k-1} \cdots\right\}$$
$$\left(L_P^{\alpha}\right) : d_j^k = d_j^k - \alpha\left(a_j^k + a_{j+1}^k\right)$$
$$\left(L_U^{\beta}\right) : a_j^k = a_j^k + \beta\left(d_j^k + d_{j+1}^k\right)$$
$$\left(L_P^{\gamma}\right) : d_j^k = d_j^k - \gamma\left(a_j^k + d_{j+1}^k\right)$$
$$\left(L_U^{\delta}\right) : a_j^k = a_j^k + \delta\left(d_j^k + d_{j+1}^k\right)$$
$$x^k[n] = \left\{\cdots x_j^k \cdots\right\}$$

(c) CDF 9/7 wavelet

Figure 5: Lifting scheme for various wavelet families.

2.1.1 Wavelet lifting schemes

The lifting scheme [118] is a well-known technique to implement a *second-generation* wavelet as a computationally-efficient successor of the *first-generation* wavelet based on a filter scheme. Figure 5 is a pictorial description of 1D DWT with a lifting scheme on several wavelet families, such as Haar, CDF 5/3, and CDF 9/7. Those wavelets have a biorthogonal basis, which allows us to have more freedom in designing a pair of analysis and synthesis functions (i.e., forward and inverse transforms). They are also *separable*, which means multi-dimensional DWT can be implemented by successively applying a 1D transform along each dimension.

Wavelet lifting schemes consist of multiple steps (e.g., two for Haar and CDF 5/3, four for CDF 9/7) with an optional normalization step at the end. First, the input signal is *split* into two streams of coefficients: *even* and *odd* arrays, or conventionally called by *approximation* and *detail* series, which indicate the prospective low- and high-frequencies of the wavelet spectrum. In other words, the multirate input signal is decomposed into a polyphase representation as follows:

$$a^k[n] = x^{k-1}[2n], \quad d^k[n] = x^{k-1}[2n+1] \tag{5}$$

where $a^k[n]$ and $d^k[n]$ are the approximation and detail coefficients of the successive transform level k from $k-1$, respectively. Next, the *detail* part is *predicted* from the current values of both streams (Equation 6) and the *approximation* part is then *updated* (Equation 7) accordingly:

$$d^k[n] = d^k[n] - L_P\left(a^k[n]\right) \tag{6}$$

$$a^k[n] = a^k[n] + L_U\left(d^k[n]\right) \tag{7}$$

Simply speaking, L_P and L_U (illustrated as the black arrow in Figure 5), which are the prediction and updating lifting operators in Equation 6 and Equation 7, can be represented as the local weighted sum of the lifting coefficients. Both Haar and CDF 5/3 lifting schemes use one pair of weights (α and β) while CDF the 9/7 scheme requires two pairs of weights, (α and β), and (γ and δ), as listed in Table 1. Finally, the resulting coefficients are sent to the appropriate locations in the *merge* step, with or without normalization, and one level of transform is completed (Equation 8). The inverse transform can be by applying the previous steps in reverse order with weights in opposite signs.

$$x^k[n] = a^k[n], \quad x^k[n + dim/2] = d^k[n] \tag{8}$$

Note that CDF 9/7 is lossy, meaning that the complete round of forward and inverse transforms does not reproduce the input signal. This is due to the non-exact lifting coefficients (see

Table 1: Lifting coefficients of selected wavelet families

	Haar	CDF 5/3	CDF 9/7
α	-1.0	-0.50	-1.58613434
β	$+0.5$	$+0.25$	-0.05298012
γ	N/A	N/A	$+0.88291108$
δ	N/A	N/A	$+0.44350685$

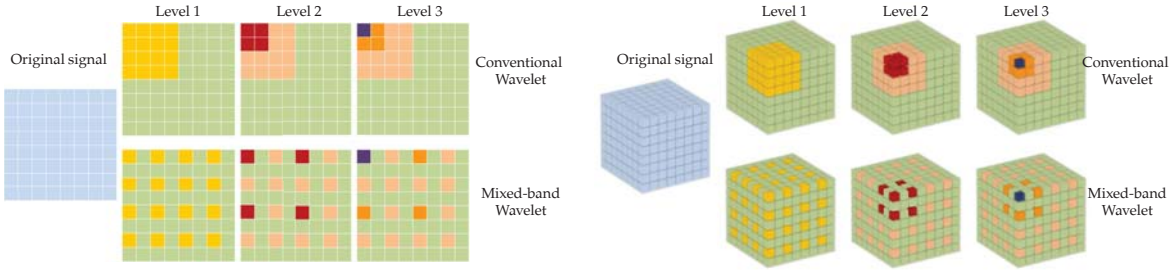


Figure 6: Memory layouts of conventional and mixed-band wavelets in (a) 2D and (b) 3D.

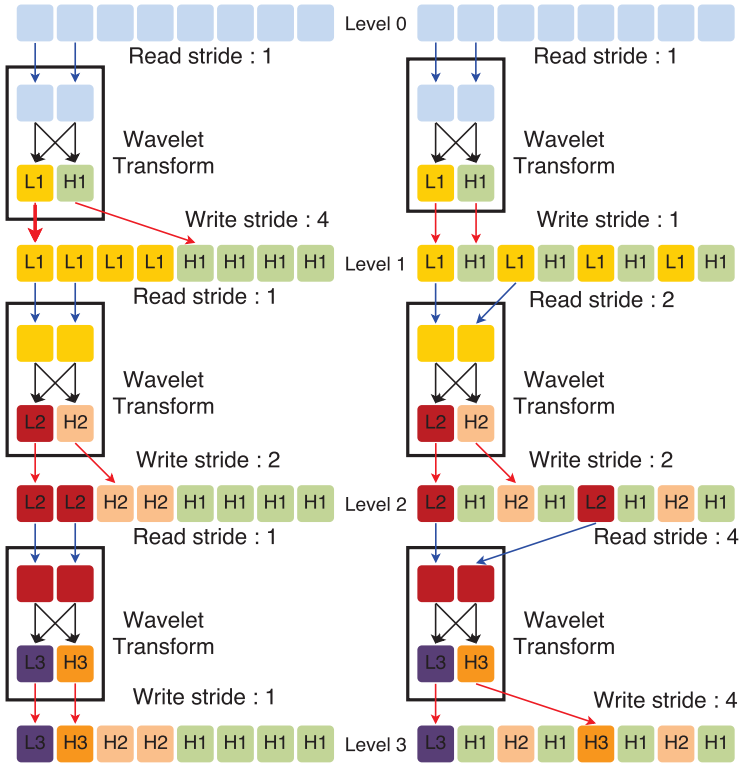


Figure 7: Comparison of memory access patterns in conventional (left) and mixed–band (right) 1D Haar wavelet transforms.

Table 1). In addition, CDF 9/7 is more computationally expensive because it requires four lifting steps, as opposed to only two steps in Haar and CDF 5/3. Figure 5 also shows that L_P and L_U of the Haar wavelet lifting transform can be done locally without reading neighboring values across the block boundary because both predicting and updating steps use a one-side support, either left or right, within a group of two pixels for a one-dimensional transform. CDF 5/3 and CDF 9/7, however, need to predict and update the current element using the neighboring values from both sides. This means that the implementation of a CDF wavelet on the GPU should follow the stencil-based computing paradigm [93] because each thread block needs to access a portion of data slightly larger than the output region, which is called *halo*. More details about block dimensions and thread assignments for handling halo will be discussed shortly.

2.1.2 Mixed-band wavelet transform

The motivation of developing a mixed-band algorithm is that the separation of different frequency bands may not be necessary for many applications as long as forward and inverse transforms work as expected. For example, the main reason that the wavelet transform is used in compressed sensing MRI reconstruction is to sparsify the image in wavelet domain and suppress close-to-zero signals. Hence, for this purpose, the location of each coefficient does not matter. Therefore, the main idea in the proposed mixed-band algorithm is to allow reading from and writing to the same memory location without rearranging the result into low- and high-frequency coefficients. This fits especially well for GPU Haar DWT because the entire single-level transform can be implemented fully *in-place* using shared memory without writing intermediate lifting results to global memory. Figure 6 shows the memory layout of wavelet coefficients in conventional (upper rows) and mixed-band (bottom rows) wavelet transforms. In the conventional DWT (top rows in Figure 6), low- and high-frequency wavelet coefficients are spatially separated (for example, in Figure 6 (a), the upper image of Level 1, the yellow region is the low-frequency coefficients and the green region is the high-frequency coefficients). In the following level, the low-frequency region of the previous level will be used as the input to the wavelet transform. In the mixed-band wavelet, however, high- and low-frequency coefficients are inter-mixed, as shown in Figure 6's bottom rows. This unconventional layout does not alter the histogram of the wavelet coefficient values, and the inverse transform can reconstruct the input image losslessly.

In order to explain the mixed-band approach in detail, without loss of generality, we use a 1D wavelet example as shown in Figure 38. Let us assume that the input is a 1D array consisting of eight pixel values. Each wavelet transform converts the input into two half-size arrays: one stores low-frequency coefficients and the other stores high-frequency coefficients. Therefore there can

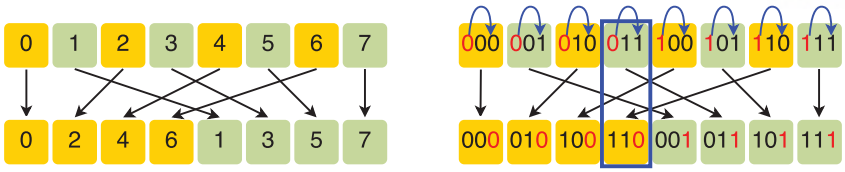


Figure 8: Bit rotation permutation on an array of 8 pixels: decimal indexing and binary indexing.

be up to $\log_2(N)$ levels when N is the size of input array (there are three levels in this example). Blue arrows are reading transactions and red arrows are writing transactions. If we assume that this process runs on the GPU, then you can consider the input and output pixel arrays as global memory and the dotted box for wavelet transform as shared memory. As shown in Figure 38 left, reading and writing strides are different in the conventional wavelet transform while those are identical in the mixed-band wavelet transform. The benefits of mixed-band approach include 1) less memory usage due to in-place filtering, 2) *fused* multi-level transformation in a single kernel call without synchronizing using global memory, and 3) reducing the index computation overhead for shuffling the location of wavelet coefficients.

2.1.3 Bit rotation permutation

In order to convert a mixed-band wavelet layout to a conventional wavelet layout, we need to recompute the index of each coefficient. Specifically, after performing the lifting step, the wavelet coefficients are divided into two groups, i.e., low- and high-frequency coefficients, as shown in Figure 8. A naive method to compute a new index value involves integer division and addition operations, which affects the performance. A 2D DWT is even more expensive because we need to manage four groups of coefficients. In order to reduce this overhead, we borrow the idea of index shuffling strategy in Fourier transform to rearrange wavelet subbands more efficiently. The fast DFT introduced in [33] uses *bit reversal permutation* to rearrange the Fourier frequency bands. Similar to this, we propose a procedure to perform fast reindexing on the wavelet spectrum, i.e., *bit rotation permutation*. Figure 8 right illustrates the 3-bit rotation to-the-right of each index on an array of 8 pixels. As shown here, the index of the target location after wavelet transform can be simply computed by rotating (or shifting) source index's bits. As a result, only bitwise operations are required to calculate output indices and the computational cost is lower. For an image of arbitrary size, the number of rotation bits can be determined by evaluating $log(\lceil dim \rceil)/log(2)$ where $\lceil dim \rceil$ is rounded to the optimal value for efficient calculation (as in DFT). This pre-calculation can be performed explicitly outside GPU kernel calls.

2.2 Multi-GPU Iterative Solver

One research direction for accelerating CS reconstruction has focused on developing efficient numerical solvers for the ℓ_1 minimization. problem [9, 51, 70]. The other direction has been leveraging the state-of-the-art parallel computing hardware, such as the graphics processing unit (GPU), to push the performance to the limit [9, 74, 94]. We believe that GPU acceleration is the most promising approach to make CS-MRI reconstruction clinically feasible, but multi-GPU acceleration has not been fully addressed in previously published literature. The main challenge in handling dynamic MRI is its size. Since the data are acquired continuously over time, there are enormous size of images need to be reconstructed at the end even though CS acquisition reduces the size of the input raw data to be processed. Moreover, if we consider 3D data, then we need to handle 4D data which ends up with handling giga– to peta–scale images to reconstruct. Although using a GPU can accelerate reconstruction time close to real–time, it is not sufficient to handle the increasing size of high–dimensional dynamic MRI. In addition, as sensitivity and spatial resolution of MRI is increasing, it is clear that larger image size will be common in the future. Therefore, employing multiple GPUs is necessary to address this problem.

2.2.1 Multi–GPU parallelization on a shared memory system

First approach is using a system having multiple GPUs connected via a PCI express bus. The system we tested on is connected to eight NVIDIA Tesla M2090 GPUs installed on a Dell C410x external enclosure. The host system is NUMA (Non–Uniform Memory Access) architecture with four six-core AMD Opteron CPUs, and four HICs (Host Interface Card) are connected to the PCI express bus where each card is connected to two GPUs. We use OpenMP [3] to split the data into multiple pieces, and distribute them across multiple GPUs to run the reconstruction algorithm in parallel. OpenMP can create several light–weight CPU threads and each thread will control one of the GPUs by using `cudaSetDevice(omp_get_thread_num())`. Since our reconstruction algorithm uses a finite difference method, we need extra neighbor pixels (i.e., halos) when split the data. We divide the data only along the time axis because usually x–y resolution is fixed but the number of slices along the time axis can vary depending on the duration of acquisition. Therefore, each GPU needs to be assigned with 1/(# of GPUs) of data and halo slices, see Figure 9.

Per each iteration, halo slices have to be exchanged between adjacent GPUs because the slices at the boundary become invalid. This data transferring is done using `cudaMemcpyPeer2Peer()`

[3] `http://openmp.org`

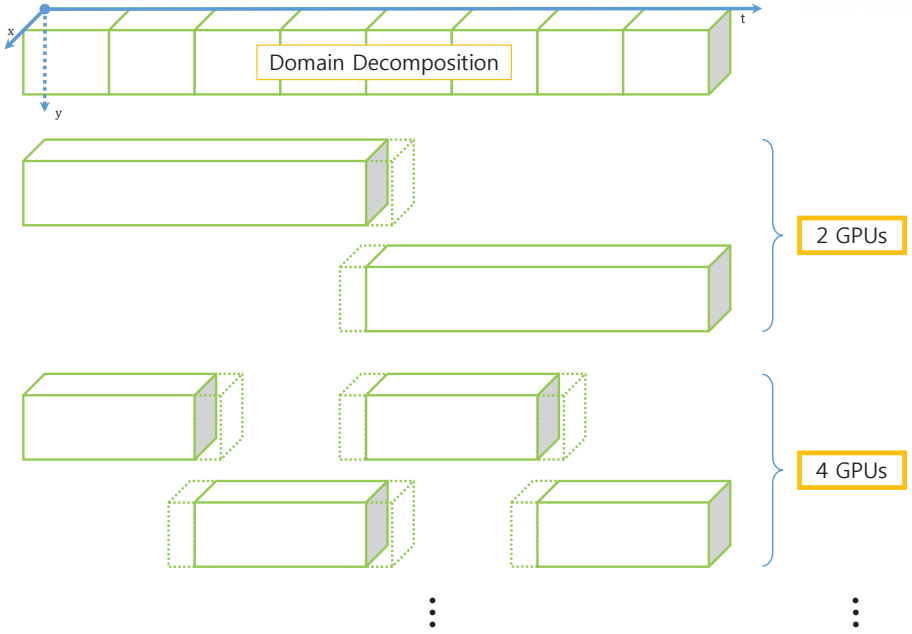


Figure 9: Data splitting based on the number of GPUs.

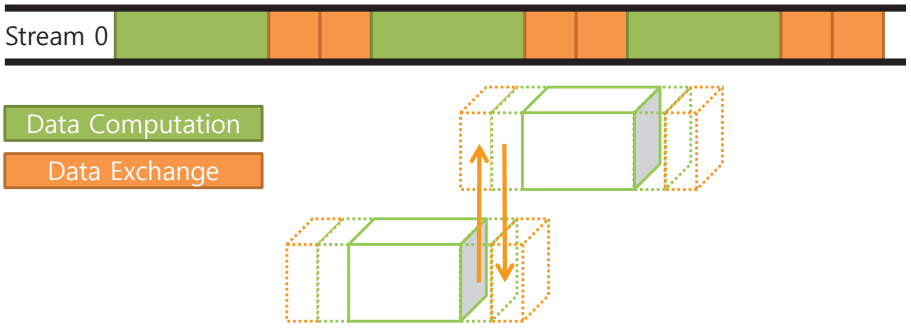


Figure 10: Halo exchange synchronously between GPUs which belong to the same node

(P2P [4] that has the maximum bandwidth roughly around ∼6 GB/s on our system. The required halo size is one slice along each direction (left, right) since we use the first–order finite difference approximation for gradient and Laplacian computation. Therefore, the GPUs in the middle should have two halo slices (one per each side), but the boundary GPU needs only one halo slice. To reduce halo communication overhead, we can group multiple slices and send them at once. In this approach, the halo size is greater than one in order to allow the method runs multiple iterations without exchanging halos with neighbor GPUs.

If we only use single stream, then all the communication and execution have to be synchronized and run serially (Figure 10). However, if we choose the halo size properly and use multiple streams, then we can hide data communication overhead by overlapping with execution. Figure 11 explains this approach where three streams are used to split execution (1 stream) and

[4] http://docs.nvidia.com/cuda/cuda-runtime-api/

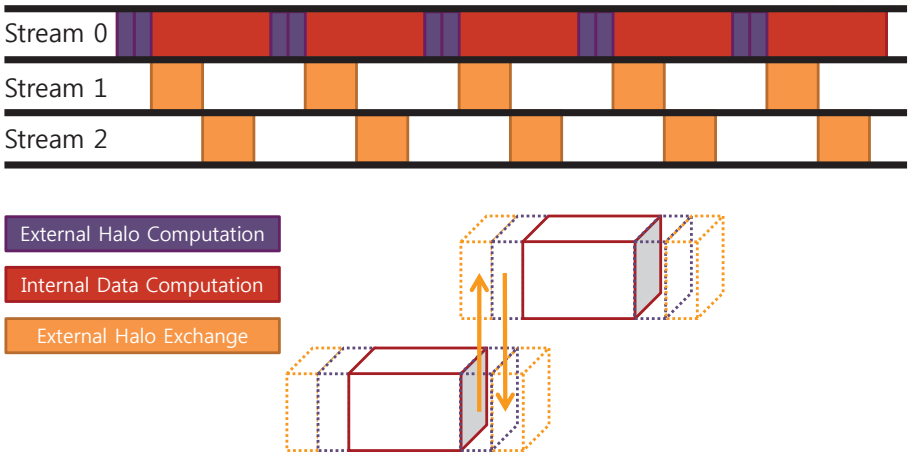


Figure 11: Hiding P2P overhead using streams by overlapping computation and communication

data communication tasks (2 streams for left and right halo). We employ the approach similar to Paulius [93] by splitting the data into three regions – external halo (orange), internal halo (purple), and internal data (red) in Figure 11. Note that external halos are used only to compute correct values for internal halos, and internal halos are exchanged with neighbor GPUs while internal data is being computed. If we choose the halo size properly (depending on the communication latency and computing cost), we can hide communication overhead by overlapping halo exchange and computation.

2.2.2 Multi–GPU parallelization on a distributed memory system

Second approach is using a system having multiple nodes connected via network, see Figure 12. In this setup, GPUs are connected to each node, and memory on each node is not visible to other nodes. The system we tested on has two nodes connected via a QDR Infiniband network, and each node has eight NVIDIA Tesla M2090 GPUs. We use MPI (Message Passing Interface) [5] to communicate between nodes. We use OpenMPI 1.7.2 [6] that supports direct communication interface between GPUs in a distributed memory system. In this setup, the data splitting strategy is same as in a shared memory system (Figure 9). However, communication between nodes is more expensive than using a PCI express bus, so we need to pay a special attention to hide communication latency effectively. We observed that the maximum bandwidth between different GPUs is roughly around ∼1.2GB/s on our system.

To communicate data between GPUs, which belong to different nodes, we have two main approaches: the conventional way and the GPU Direct–enabled way. In the conventional one,

[5] http://www.mpi-forum.org/

[6] http://www.open-mpi.org/software/ompi/v1.7

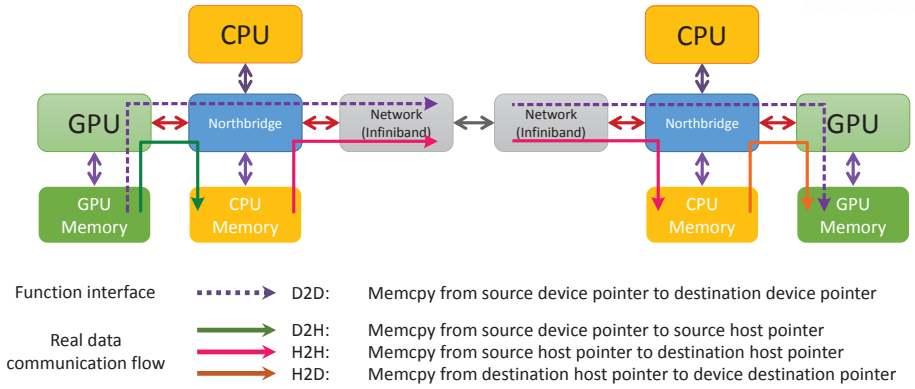


Figure 12: GPU direct version 2.0, function interface and the actual data communication flow

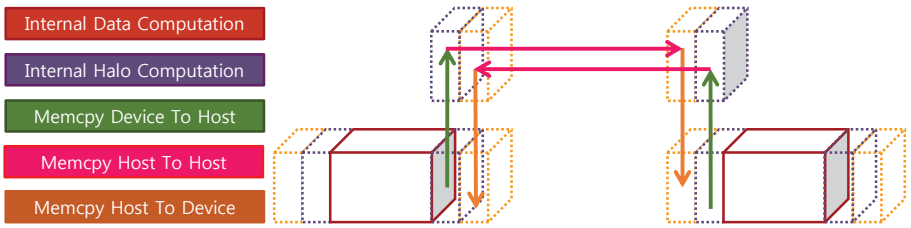


Figure 13: Halo exchange between GPUs which belong to the different nodes

data from the valid regions firstly need to be copied back from GPU to CPU memory (device to host copy). Then, we use MPI communication functions to transfer those halos to the neighbor CPU node. Lastly, the neighbor node needs to send data to their appropriate GPUs (host to device copy). Figure 13 describes this flow. This method serves the naive attempt when the communication between nodes are blocking (i.e., `MPI_Send(...)`, `MPI_Recv(...)`). Although, copy the data from device to host and vice versa are asynchronous, we still have to wait when the first halo is completely sent, then continue the second halo. As shown in Figure 14, the GPUs are idling before receiving the valid data in order to proceed the next iteration.

We can further improve this approach by switching to the non–blocking (i.e., `MPI_Isend(...)`, `MPI_Irecv(...)`) version. In this case, the CPU process immediately returns to the ready stage to send the second halo. Therefore, we can save a lot of time and keep the GPU compute engines busy. Figure 15 shows that the big gap between two computing blocks on the main stream has been reduced a lot. During the optimization process, we observe that the left and right halo buffer in host memory should be separated by different pointers. That means, we can not create a large buffer that contains interior halos to be sent and exterior halos to be received by using only one pointer and a stride from that pointer in the same buffer because non–blocking communication will recognize this as transferring from the same memory location and enforce synchronous communication. In addition, either blocking or non–blocking imple-

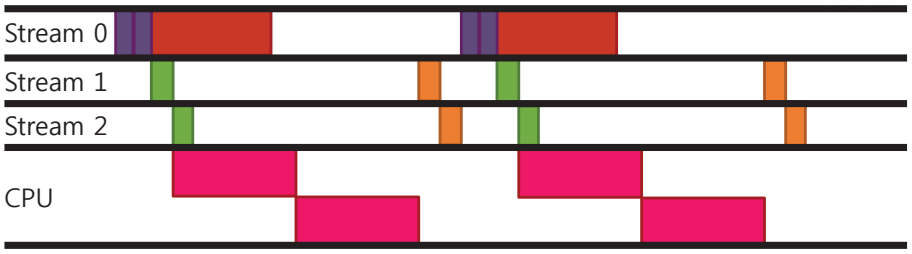


Figure 14: Overlap data communication between distributed GPUs, a blocking approach

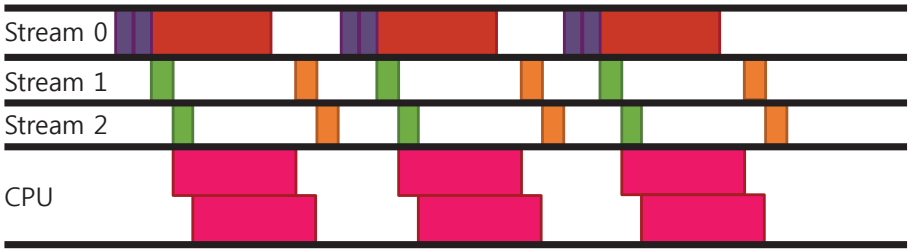


Figure 15: Overlap data communication between distributed GPUs, a non-blocking approach

mentations should have sophisticated synchronizations, such as `cudaEventSynchronize(...)` for device and `MPI_Wait(...)` for host, to ensure that data transfers have been completed before resuming the iteration jobs.

Another approach, which is GPU Direct–enabled method, is also worth considering, see [7] for more details. In brief, we have several advantages by utilizing this technique. Version 1.0 provides a fast communication between GPU memories from different nodes, user can send the data from this device to another device with only once couple invoking `MPI_Send(...)`and `MPI_Recv(...)`. The MPI low–level of implementation will do the rest things which are the same as the conventional approach. The only slightly difference is the memory copy from device to host and host to device are completely *synchronous*. This introduces a performance bottleneck although it splits the data into many small chunks and transfers them in a pipeline fashion. Version 2.0 provides an improvement for shared memory systems. If two MPI processes running on two GPUs are belong to the same node, `MPI_Send(...)`and `MPI_Recv(...)` can automatically call `cudaMemcpyPeer(...)` to have a direct access between those device memories and the communication time can be reduced. Although our GPU Cluster has Fermi GPUs and can not use GPU Direct version 3.0, it is worth mentioning here that GPUs from different nodes can communicate directly without coordination of the CPU by using Remote Direct Memory Access (RDMA). That means, halo data can directly go to chipset and then transfer via the network protocol.

[7] `http://developer.nvidia.com/gpudirect`

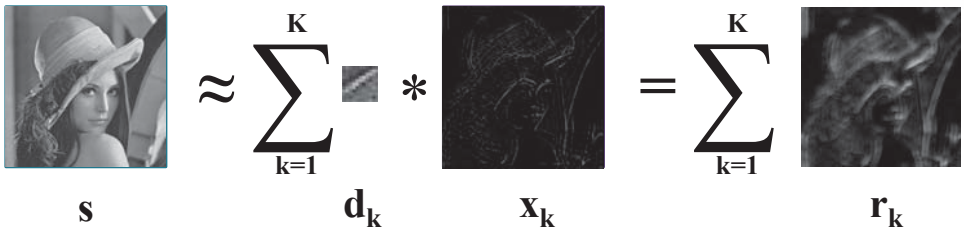


Figure 16: Decompose an image into a collection of *response maps*.

2.3 Convolutional Sparse Coding Solver

One of the main problems hinders adopting MRI for time-critical application is its longer acquisition time. There has been much research effort to accelerate MRI acquisition process using hardware and software. Among them, compressed sensing has been successfully endorsed as a software approach to reconstruct high-quality images from undersampled raw MRI data (i.e., k-space data). Since CS-MRI imposes an additional computation burden and suffers from reconstruction artifact, CS-MRI research has mostly focused on developing faster numerical algorithms and improving image quality for low sampling rates. The earlier CS-MRI work mainly focused on the ℓ_1-norm energy minimization problem over sparse signal generated using universal sparsifying transforms, such as Total Variation, Wavelet, and Fourier transform, as introduced in the seminal work by Lustig et al. [86]. Even though such methods may be relatively fast due to their simplicity, the image quality may not be optimal because such universal transforms may not represent various local image features effectively. GPU-acceleration has also been well-adopted to reduce the computing time of such algorithms [99]. On the other hand, recent data-driven approaches, such as dictionary learning [1], are adopted to CS-MRI reconstruction and showed significant improvement in image quality [17]. The main novelty of this approach is to derive a sparsifying transform via a machine learning process, which accurately represent local features of reconstructed image compared to those of universal transforms. However, a drawback of this approach is its computational cost because patch-based dictionary learning is a highly time-consuming process. A recent advance in dictionary learning, convolutional sparse coding [15], replaces the patch-based dictionary learning process with an energy minimization process using a convolution operator on the image domain, which leads to an element-wise multiplication in frequency domain, derived within ADMM framework [12]. Later, a more efficient method based on a direct inverse problem is proposed by Wohlberg [131]. However, such advanced machine learning approaches have not been fully exploited in CS-MRI literature yet.

Figure 17: A 64-atom dictionary generated from natural images.

CSC can be regarded as an alternative to dictionary learning [1], which builds the dictionary with convolution filters instead of local patches. In this section, we present a brief background review of CSC for 2D images. For a given image s, we would like to find its best approximation from the summation of *response maps* $\sum r_k$. This reverse problem can be solved if we impose a constraint that each *response map* r_k is the result of convolution between a *filter* (or *atom*) d_k and its associated *sparse map* x_k. The non-linear ℓ_p norm (where $0 \leqslant p \leqslant 1$), which is applied to x_k (i.e., x_k is sparse), affords a feasible solution of finding such a collection of d_k and x_k. The term *sparsity* is well-known in compressed sensing [37] area – for a given particular signal x_k, it is said to be sparse if most of elements in x_k are close to zero. Figure 16 is a pictorial description of finding d_k and x_k in CSC problem. As can be seen, most of the pixels in the *sparse maps* x_k are zeros (black), and there are only a few of non-zero valued (white pixels). Once d_k and x_k are obtained properly, we can compute back the *response map* r_k by convolving the corresponding *filter* and *sparse map*.

Mathematically, CSC problem is equivalent to minimize this energy function:

$$\min_{d,x} \ \frac{\alpha}{2} \left\| s - \sum_k d_k * x_k \right\|_2^2 + \lambda \sum_k \|x_k\|_1 \quad s.t.: \quad \|d_k\|_2^2 \leqslant 1 \tag{9}$$

where d_k is the k-th filter (or atom in the dictionary) and x_k is its corresponding sparse map for s. In Equation 57, the first term measures the difference between s and its sparse approximation $\sum_k d_k * x_k$, weighted by α. The second term is the sparsity regularization of x_k using an ℓ_1 norm with a weight λ instead of an ℓ_0 norm as used in [1]. The remaining constraint restricts the Frobenius norm of each atom d_k within a unit length.

If we wish to train d_k to generalize for the dictionary and to represent the features of many images obtained from a database, all the training instances can be fetched and participate in contributing to the feature extraction. For example, Figure 17 shows a 64-atom dictionary that

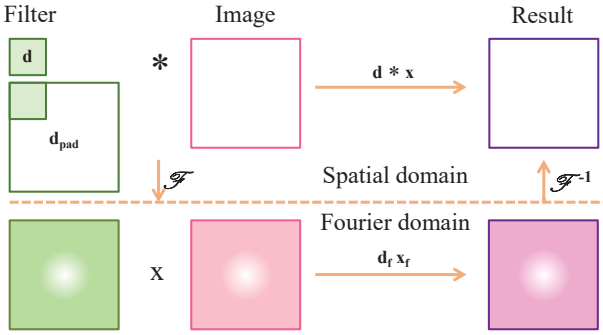


Figure 18: Convolution in image domain equals pixel-wise multiplication in frequency domain.

has been trained using a collection of natural and standard images (lena, barbara, etc.). Their components in Gabor-like shapes capture directional edges that match the fundamental features in our human visual perception.

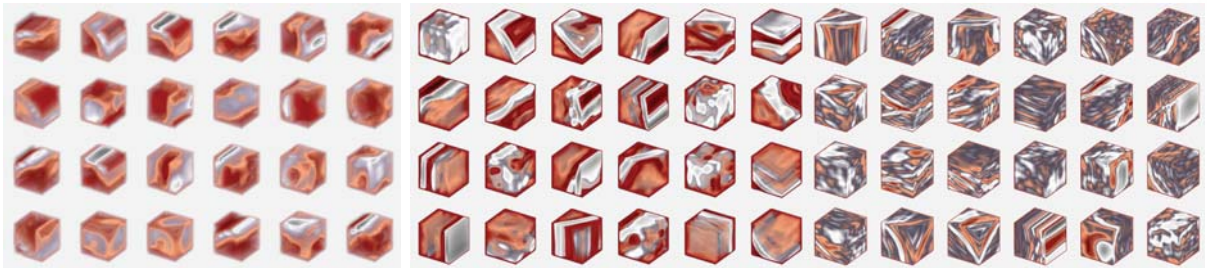
Solving Equation 57 is introduced in the seminal work of [137]. They proposed an alternating strategy in which a series of convex subproblems between d_k and x_k is solved until convergence. Because their solver is completely in the image domain, the linear complexity of convolution affects the performance of their algorithm. More efficient approaches based on the Fourier Convolution theorem are also proposed [15, 131, 132] (see Figure 18). The filter is padded with zeros to make it the same size as the image, and the Fourier transform is applied to both the padded filter and the image so that the convolution can be computed as a pixel-wise multiplication in the Fourier domain.

Dictionary can also be in 3D format. As shown in Figure 19, 3D dictionaries are extracted from the input volumes where each atom in the dictionary represents local feature in the data collected from public domain: MRI-Kiwi, MRT-Aneurysm, CT-Bonsai, low-dose CT-Chest (LdCT-Chest), CT-Tooth, and EM-Mouse. For example, the atoms in the dictionary of the LdCT-Chest data represent fine-level details related to bones and surrounding muscle tissues. Similar results can be found in other datasets as well. They generate the data-dependent learned basis, which differs significantly from the universal dictionary in the Fourier, cosine and wavelet transforms.

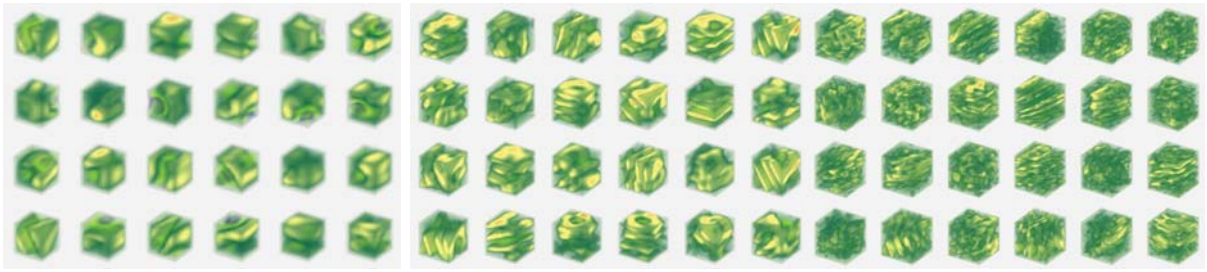
2.4 Deep Learning in Bio-medical Image Reconstruction

2.4.1 Deep Learning in Image Processing and Computer Vision

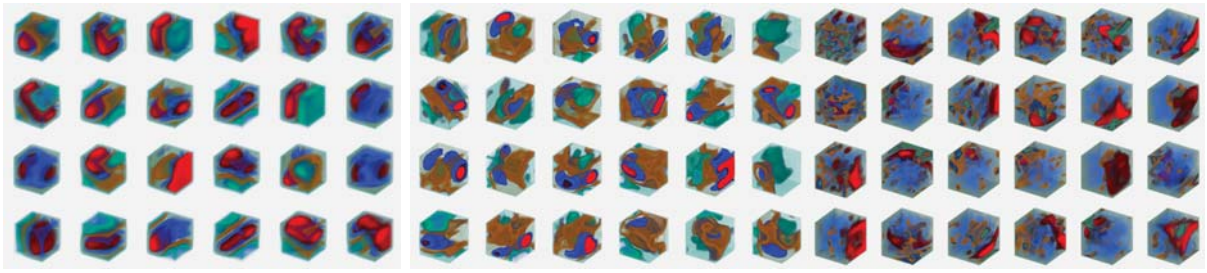
In the last five years, deep learning [78] has gained much attention, largely because it has surpassed the human level in solving many complex problems. It is comprised of many perceptron layers that form a deep neural network. In visual recognition tasks, this type of architecture can learn to recognize patterns such as handwritten digits and other features of interests [77] in



(a) Dictionaries of LdCT-Chest dataset



(b) Dictionaries of MRI-Kiwi dataset



(c) Dictionaries of CT-Bonsai dataset

Figure 19: Volume rendering of hierarchical multi-scale 3D dictionaries. From left to right: three resolutions of 24 atoms (7^3, 15^3 and 31^3).

images hierarchically [136]. However, the main drawback of using deep neural network is that it requires a huge amount of data for training the network. In order to overcome this issue, researchers have started to collect a large database [107] which contains millions of images from hundreds of categories. Since then, many advanced architectures have been introduced including VGG [111], Googlenet [119]. Computers are now able to mimic artistic painting to produce new pictures by transferring the style from one image to another [49]. In addition, researchers are also actively working on extending deep learning methods for medical image data beyond the scope of natural images [30]. These approaches impose vast changes in automatic classification and segmentation on other image modalities, such as CT [138] and MRI [66].

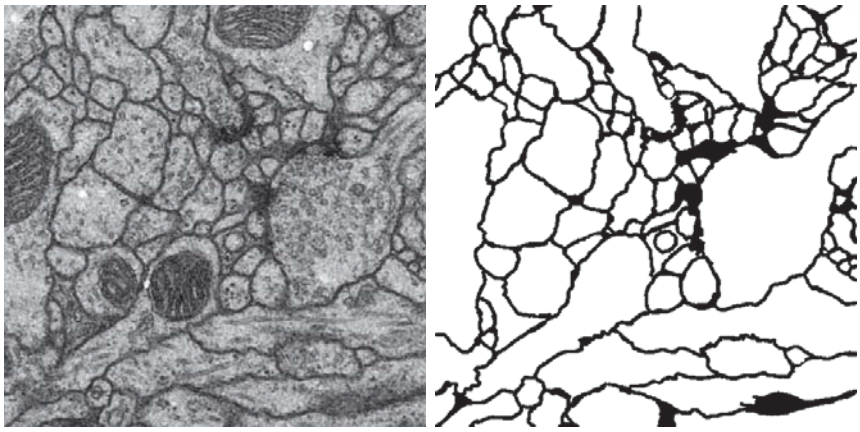


Figure 20: An example of EM image (left) and its cell membrane segmentation result (right).

2.4.2 Deep Learning in Connectomics

"How does the brain work?" This question has baffled biologists for centuries. The brain is considered the most complex organ in the human body, which has limited our understanding of how relating its structure is related to its function even after decades of research [83]. Connectomics research seeks to disentangle the complicated neuronal circuits embedded within the brain. This field has gained substantial attention recently thanks to the advent of new serial-section electron microscopy (EM) technologies such as the automated tape-collecting ultramicrotome (ATUM) [55] (see Figure 20 for an example of EM image and its cell membrane segmentation). The resolution afforded by EM is sufficient for resolving tiny but important neuronal structures that are densely packed together, such as dendritic spine necks and synapses. These structures can be as small as only tens of nanometers in width [59]. Such high-resolution imaging results in the generation of enormous datasets, approaching one petabyte for only a relatively small tissue volume of one cubic millimeter. Therefore, handling and analyzing the resulting datasets is one of the most challenging problems in connectomics.

Early connectomics research focused on the sparse reconstruction of neuronal circuits [11, 14], i.e., tracing only a subset of neurons in the data by using manual or semi-automatic tools [23, 69, 113]. Unfortunately, this approach requires too much human interaction to scale well over the vast amount of EM data that can be collected with technologies such as ATUM. Because of this, the field has been limited in the number of datasets that have been thoroughly annotated and analyzed. In addition, multi-scale reconstruction, including dense reconstruction in the region of interest, has gained popularity recently because it can reveal low-level structural information

that is not available in sparse reconstruction or functional imaging [73]. Therefore, developing scalable and automatic image analysis algorithms is an important and active research direction in the field of connectomics. Although some EM image processing pipelines (e.g., RhoANA [73]) use conventional, light-weight pixel classifiers, the majority of recent automatic image segmentations for connectomics rely on deep learning. Earlier automatic segmentation work using deep learning has mainly focused on patch-based pixel-wise classification based on a convolutional neural network (CNN) for affinity map generation [124] and cell membrane probability estimation [31]. However, one limitation of applying a conventional CNN to EM image segmentation is that per-pixel network deployment can be highly expensive in consideration of the tera- to peta-scale EM data size. For this reason, a more efficient, scalable deep neural network will be important for image segmentation of the large datasets that can now be produced [28, 105]. The main idea behind these approaches is to extend a fully convolutional neural network (FCN) [85], which uses encoding and decoding phases similar to an autoencoder for the end-to-end semantic segmentation problem.

Deep learning has been quickly adopted by connectomics research for automatic EM image segmentation. One of the earliest applications to EM segmentation was made by Ciresan et al. [31]. This method involves the straightforward application of a CNN for pixel-wise membrane probability estimation and it won the ISBI 2012 challenge [3]. One notable recent advancement in the machine learning domain is the introduction of a fully convolutional neural network (FCN) [85] for the end-to-end semantic segmentation problem. Inspired by this work, many successive variants of FCN have been proposed for EM image segmentation. Chen et al. [28] proposed multi-level upscaling layers and their combination for final segmentation. A new-post processing step, namely lifted multi-cut [5], was also introduced to refine the segmentation. Ronneberger et al. [105] presented skip connections for concatenating feature maps in their U-net architecture. Although U-net and its variants can learn multi-contextual information from the input data, they are limited in the depth of the network they can construct because of the vanishing gradient problem. Recently, the 3D extension of U-net was proposed for confocal microscopy segmentation [30]. In the image classification task, on the other hand, shortcut connections and direction summations [57] allow gradients to flow across multiple layers during the training phase. Overall, these related studies inspired us to propose a fully residual convolutional neural network for analyzing connectomic data. Work that leverages recurrent neural network (RNN) architectures can also accomplish this segmentation task [116]. In fact, the membrane-type segmentation approach is a crucial step for connected component labeling to resolve false splits and merges during the post-processing of probability maps [41, 96].

Modern electron imaging techniques enable comprehensive reconstruction of neural circuits at nanometer resolution. However, electron microscopy images often have limited contrast, which can make identification of features ambiguous. In cases where we can model sources of noise, it may be possible to denoise the data, improving effective signal-to-noise ratios and improving both manual tracing and automated segmentation accuracy. Conventional approaches do not usually harness the noise-free samples as ground truth to perform image denoising task. Several common filter design techniques can be named as a few, such as Bilateral filter [122], Non-Local Mean filter [16], so on and forth. On the other hand, noise reduction problem can be also defined as an optimization scheme where Total Variation (TV) [26, 106] either in ℓ_1 or ℓ_2 norms of the results as regularizers, Anisotropic Diffusion (Perona–Malik Diffusion) [97] are proposed . Although those methods are less susceptible to the computational burden where intensive data and heavy training are required, they posed some difficulties for choosing the best parameters via tuning. Modeling image as sparse linear combination of atoms, such as K-SVD [39], BM3D [34] is another category of denoising techniques that can robustly remove the noise. A group of atoms, i.e., dictionary, after being trained offline (with clean data) or blindly online, can be used to estimate the noisy image in which noise-model can not be captured via sparsity regularizer and results in the noise-free approximation. However, these methods pose computational expense in both training and estimating the denoised images in which many iterative minimization steps are involved for solving a predefined energy functions. Deep learning [52, 78] has emerged recently when people made advancements and applied them to various tasks of image processing and computer vision problem such as classification [58, 65, 77], segmentation [85, 105], localization [50, 56], translation [67, 140], reconstruction [54, 103], and many more. Among them, cycle consistent loss has been used to denoise multiphase Coronary CT by direct translating low-dose to routine-dose CT image [72]. Hence, we can increase the image quality further by leveraging deep learning power to perform such a reconstruction task like denoising.

2.4.3 Deep Learning for Compressed Sensing MRI Reconstruction

Deep learning-based CS-MRI is aimed to design fast and accurate method that reconstruct high-quality MR images from under-sampled k-space data using multi-layer neural networks. Earlier work using deep learning in CS-MRI is mostly about the direct mapping between a zero-filling reconstruction image to a full-reconstruction image using a deep convolutional autoencoder network [128]. Lee *et al.* [79] proposed a similar autoencoder-based model but the method learns noise (i.e., residual) from a zero-filling reconstruction image to remove undersampling artifacts. Another interesting deep learning-based CS-MRI approach is Deep ADMM-Net [117], which is a

deep neural network architecture that learns parameters of the ADMM algorithm (e.g., penalty parameters, shrinkage functions, etc.) by using training data. This deep model consists of multiple stages, each of which corresponds to a single iteration of the ADMM algorithm. Recently, Generative Adversarial Nets [52] (GANs), a general framework for estimating generative models via an adversarial process, has shown outstanding performance in image-to-image translation. Unsupervised variants of GANs, such as DiscoGAN [75] and CycleGAN [140], have been proposed for mapping different domains without matching data pairs. Inspired by their success in image processing, GANs have been employed for reconstructing zero-filling under-sampled MRI with [91] and without [134] the consideration of data consistency during the training process. As shown above, deep learning has proven itself very promising in CS-MRI as well for reducing reconstruction time while maintaining superior image quality. However, its adaptation in CS-MRI is still in its early stage, which leaves room for improvement.

III GPU-based DWT with Hybrid Parallelism

3.1 GPU Optimization stategies

In this section, various optimization strategies for GPU implementation of lifting DWT are discussed in detail, and an optimal implementation strategy using hybrid parallelism will be proposed at the end.

3.1.1 Using shared memory

Shared memory is commonly used in GPU programs to reduce the longer memory latency of global memory (DRAM), and it is effective if data is reused multiple times. Another important role of shared memory is to serve as a communication medium between threads. The lifting scheme requires multiple computation steps, in which each step updates only a half of the input data (either even- or odd-indexed data), and newly updated data is used as an input to the next update step. Figure 22 (a) shows CDF 5/3 lifting wavelet implementation using shared memory for interthread communication. In the figure, red arrows represent shared memory read/write transactions and gray dotted arrows represent no memory transaction. Because only even- or odd-indexed values are updated concurrently at any given lifting step, there is no read-write conflict. Shared memory is even more effective for wavelets with large supports, such as CDF 9/7, because many update iterations must be performed.

Special care should be taken when data is copied from global memory to shared memory, since global memory access is an expensive operation on the GPU. Most wavelet bases other than Haar have a large support that extends beyond the computing domain, which is called *halo*. Due to halo, the tile dimension is not aligned with that of the thread block. For example, as shown in Figure 21, for a {32, 16} thread block, the required tile size including the halo for CDF 5/3 will be {35, 19} because the halo size along one axis is 3 (2 for left/top, and 1 for right/bottom). In order to load the entire tile region from global memory, we first linearize 2D thread indices to 1D, and then assign threads to appropriate global memory locations in a row-major order (Figure 21). Because our tile dimension is not a multiple of 128 Bytes (L1 cache line size), we disable L1 cache and use the non-caching global memory load to reduce the memory transaction granularity to 32 Bytes. The entire tile can be copied from global to shared memory in two loading steps for 512 threads (only 153 threads participate the data copy in the second step).

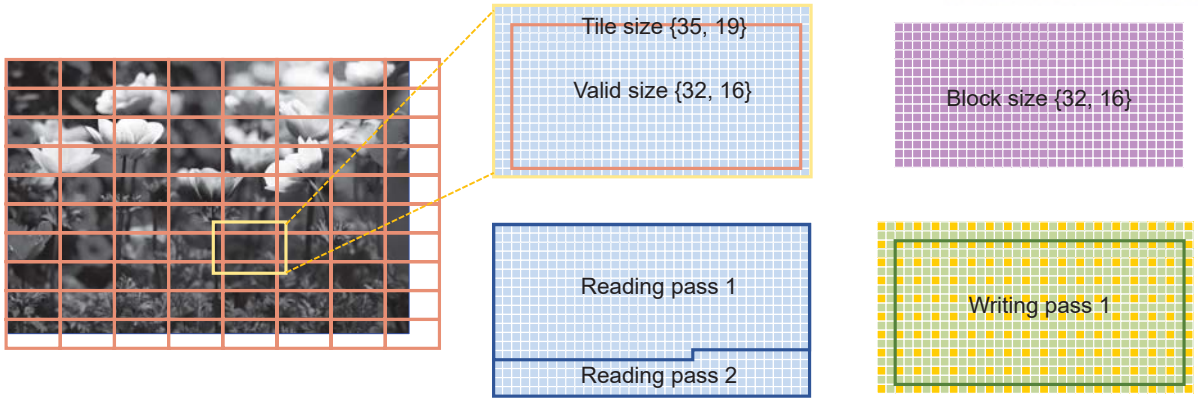
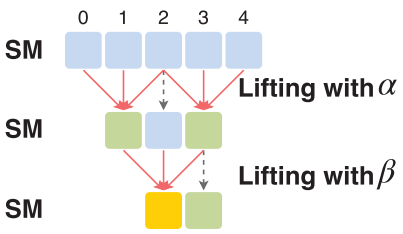


Figure 21: Region of reading and writing on using only shared memory strategy.

3.1.2 Using registers

In this optimization, we reduce shared memory transactions by loading necessary neighboring values to registers and perform computation only on registers. The main idea behind is that register's memory bandwidth is much higher than that of shared memory even though both are on-chip memory [127], so reducing shared memory transactions is beneficial although more registers are required.

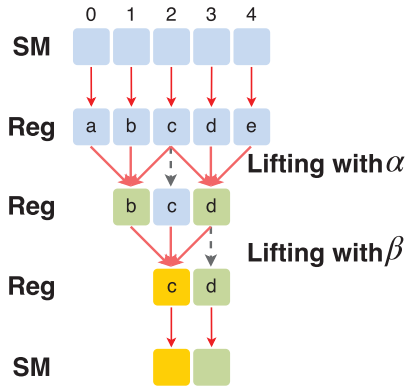
Figure 22 (a) is a shared memory-based CDF 5/3 implementation, in which in-place computation of the lifting scheme is done via writing back and forth from shared memory. This is a commonly used technique for interthread communication, but we can optimize even further by using registers as shown in Figure 22 (b), in which registers are used as a temporary per-thread buffer to hold the result of each lifting step computation. As shown in the illustration, the total number of shared memory transactions (drawn in red arrows) is reduced in the register-based implementation. The shared memory-based implementation shown in Figure 22 (top) (a) requires 6 reads (two sets of three reads in lifting with α are done in parallel), 3 writes to shared memory and two `__syncthreads()` calls. But the register-based implementation shown in Figure 22 (bottom) requires only 5 reads and 2 writes of shared memory per thread with only one `__syncthreads()` call at the end. Although there is extra data copy between registers in the register-based implementation, we observed an overall performance improvement over shared memory-based implementation about 20%. Note that the strategy described here does not exploit instruction level parallelism (ILP) yet. It can further improve the performance as shown in the following sections.




```
if((threadIdx.x&1)==0) //Encode lifting with alpha
    sharedMem[at(index_2d.x + 1, index_2d.y)]
    -= alpha*(sharedMem[at(index_2d.x + 0, index_2d.y)]
              +sharedMem[at(index_2d.x + 2, index_2d.y)]);
__syncthreads();

if((threadIdx.x&1)==0) //Encode lifting with beta
    sharedMem[at(index_2d.x + 2, index_2d.y)]
    += beta *(sharedMem[at(index_2d.x + 1, index_2d.y)]
              +sharedMem[at(index_2d.x + 3, index_2d.y)]);
__syncthreads();
```

(a) Shared memory-based approach




```
if((threadIdx.x&1)==0){
    a = sharedMem[at(index_2d.x + 0, index_2d.y)];
    b = sharedMem[at(index_2d.x + 1, index_2d.y)];
    c = sharedMem[at(index_2d.x + 2, index_2d.y)];
    d = sharedMem[at(index_2d.x + 3, index_2d.y)];
    e = sharedMem[at(index_2d.x + 4, index_2d.y)];
    //Encode lifting with alpha
    b -= alpha*(a+c);
    d -= alpha*(c+e);
    //Encode lifting with beta
    c += beta *(b+d);
    sharedMem[at(index_2d.x + 2, index_2d.y)] = c;
    sharedMem[at(index_2d.x + 3, index_2d.y)] = d;
}
__syncthreads();
```

(b) Register-based approach

Figure 22: Two lifting scheme implementations of biorthogonal CDF 5/3 wavelet and their NVIDIA CUDA code snippets.

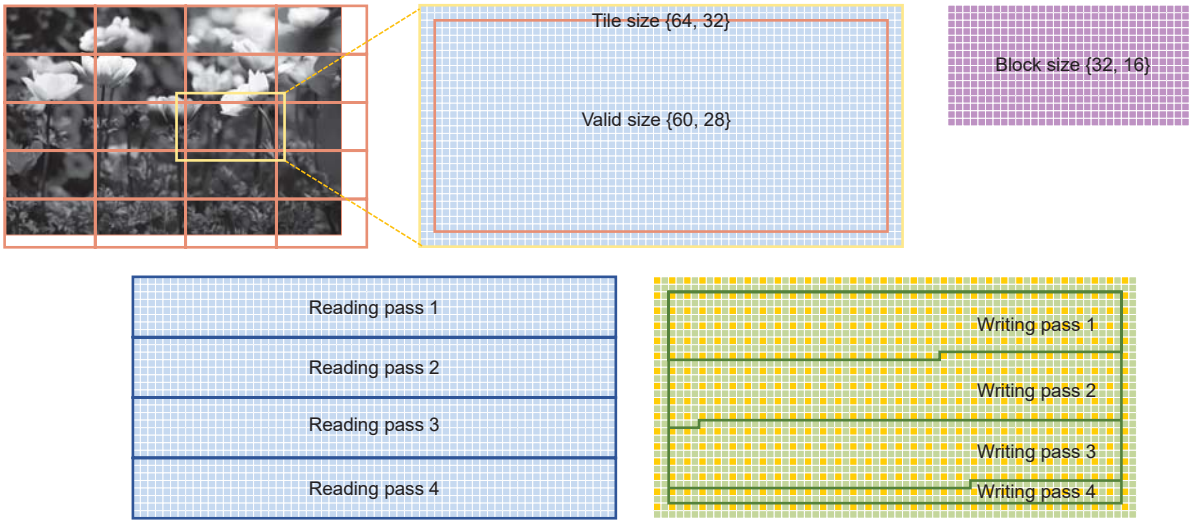


Figure 23: Region of reading and writing on using instruction level parallelism strategy.

3.1.3 Exploiting Instruction level parallelism (ILP)

In the previous optimization strategies, about a half of GPU cores are idling during DWT lifting steps. We can further reduce that inefficiency by leveraging ILP, which reduces the number of threads and let each thread do more work. For example, if we reduce the thread block size by half, then the thread block can cover the same output region without idling by doubling each thread's workload. Therefore, all the GPU cores will participate in each even- or odd-lifting step.

The ILP setup for CDF 5/3 is shown in Figure 23 where the thread block size is {32, 16} and the tile size, which is equivalent to the shared memory size, is {64, 32}. In this setup, the effective output size becomes {60, 28} due to the halo of size 2 on each side. Note that in this setup we match the shared memory dimension to a multiple of thread block dimension, which is different from the shared memory strategy. Each thread covers a {2, 2} region, which is 4× more memory transactions and 2× more lifting computations per thread. For the CDF 9/7 DWT case, we can keep the same thread block and tile dimension by reducing the output size to {56, 24} because the halo size is 4 on each side. The thread block size is then tuned empirically by fixing the x dimension to 32 and testing various y dimensions from 2 to 60. In our experiment, 20 produced the best result, i.e., 20 warps per a thread block.

Note that in this approach we did not use a register technique. Even so, we observed about 5× performance boost compared to the shared memory-based approach introduced in Section 3.1.1, which is due mainly to hiding memory and instruction latency and reducing unnecessary idling in the lifting steps.

3.1.4 Exploiting warp shuffles on Kepler GPUs

Recent NVIDIA GPUs (Kepler and later) support warp shuffle instructions, which allow for the rapid exchange of data between threads in a same warp. Currently, four warp shuffle instructions are provided: `__shfl_up`, `__shfl_down`, combined with `__shfl_xor`, and `__shfl`. Using warp instructions, a thread can directly read another thread's register values without explicitly exchanging them via shared memory. In the shared memory-based strategy introduced above, each lifting step was required to write the intermediate result back to shared memory so that neighboring threads can use it for the next lifting step. Therefore, by organizing the tile size as a multiple of warp size and declaring registers to hold the intermediate pixel values, we can eliminate effectively the expensive synchronization which can happen on shared memory in lifting steps.

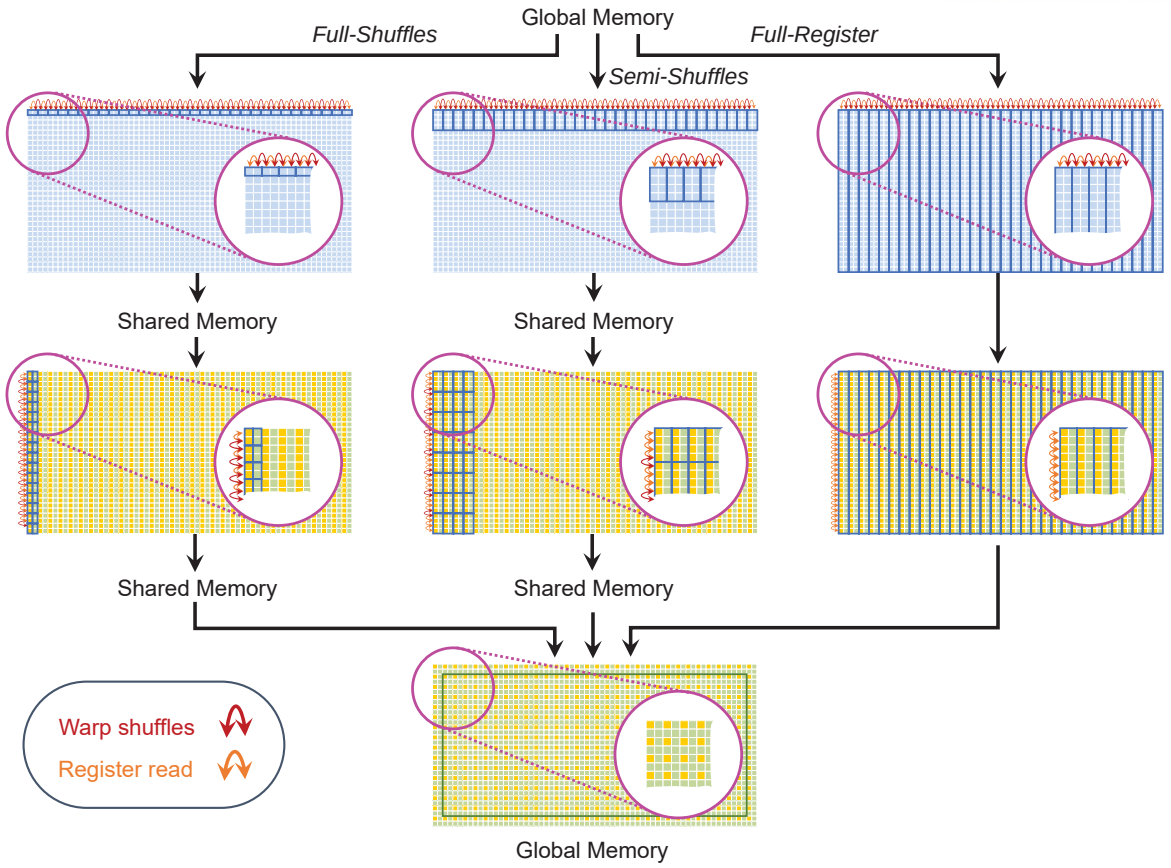


Figure 24: Three variants of hybrid parallelism of DWT: *Full-Shuffles*, *Semi-Shuffles* and *Full-Register*.

In order to exploit warp shuffles, we need to align the warp and the lifting direction, e.g., a horizontal warp for horizontal lifting step (Figure 24 left). In addition, the y dimension of the tile size should be a multiple of a warp size so that a warp can process columns without thread idling. In our setup, the tile size is {64, 32} and the thread block size is {32, 32}. As shown in Figure 24, a warp processes one row for a horizontal lifting and two columns for a vertical lifting so that each thread is processing {2, 1} output regions. In each lifting step, *every* thread must read one register value from its neighboring thread using a warp shuffling instruction, so we call this strategy *Full-Shuffles*. Note that this warp shuffle can eliminate shared memory access during lifting steps, but all the data must be written back to shared memory twice: one is between horizontal and vertical lifting and the other is immediately before the final output is written to global memory. The former instance is for inter-warp communication, and the latter instance is for coalesced global memory writes.

3.1.5 Combining all: hybrid approach

In this hybrid approach, we combine all the previously discussed techniques, such as leveraging shared memory, registers, warp shuffles and ILP to maximally hide the latency of memory operations. This method declares more registers per thread, e.g., an array of 4 by 2, and uses a {32, 8} thread block (8 warps) to cover a {64, 32} tile. Hence, there are 8× of memory operation-ILP (MILP) and 4× of computation-ILP (CILP). Regarding the lifting steps during the horizontal pass, it operates similar to *Full-Shuffles*. The slight difference is that each thread must perform DWT 4× the amount of work compared to *Full-Shuffles*. Thereafter, the intermediate result of the horizontal transform is written back into shared memory so that warps can later access the vertical pixels in aligned orders.

Subsequently, the lifting steps along the vertical pass will take place as a combination of registers and warp shuffles. The computation, which happens inside each thread, requires register-read only (Figure 24 center, orange arrows). At the top or bottom of the thread's array, neighboring pixel values must be fetched via warp shuffles (Figure 24 center, red arrows). This configuration is called *hybrid* or *Semi-Shuffles*. In fact, the degree of ILP can be modified by adjusting the *number of warps per tile*. Further, when the number of warps per tile is reduced to one and each thread holds enough registers to proceed lifting steps vertically (an array of 32 by 2), this will be similar to the setup used by Enfedaque et al. [40]. This particular setup is named *Full-Register* (Figure 24 right). Note that in *Full-Register*, shared memory is completely removed and more warps can be grouped to form a single block.

3.1.6 Fused multi-level Haar DWT

As mentioned earlier, Haar DWT has only one-side local supports and its lifting-scheme is completely fit onto CUDA blocks because halos across the block boundary are not needed. If we modify the memory layout of the conventional Haar DWT to that of the mixed-band approach, multiple levels of Haar wavelet can be processed in a single kernel call, i.e., *fused* multi-level transformation. The main idea behind this is that the mixed-band approach allows Haar DWT to be processed in-place, where a group of four pixel is read, lifted and written onto the same memory location. Therefore, once a chunk of processing block is loaded to shared memory, wavelet transform can be applied on shared memory iteratively without writing intermediate results back to global memory. The first level of transform takes place as normal mixed-band Haar DWT where implementation is exactly same as the hybrid approach except we do not reshuffle the location of wavelet coefficients. In the following levels, the method simply reads

the appropriate coefficients from shared memory, performs lifting steps, and stores the result back to shared memory. Note that ILP will decrease after each level because less number of coefficients need to be processed. In addition, there is a limit for the tile size per block, so the maximum DWT level should be chosen appropriately. In our implementation, we chose four levels of transformation to be fused into a single kernel call.

3.2 Result

3.2.1 Comparison of various strategies

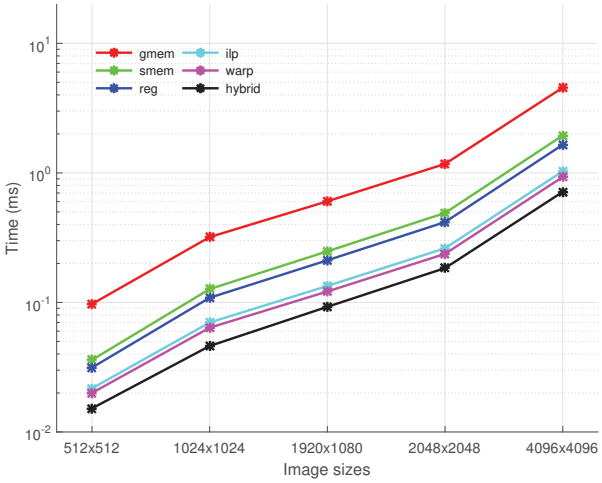
We verified the performance of the proposed method by performing one level of 2D DWT on images of different sizes. We selected a computer equipped with an NVIDIA GK110 GPU to collect the running times for all measurements. Note that in this evaluation, all of the pre- and post-transfers from CPU to GPU (and vice versa) were omitted (only the kernel timing was taken into account). We conducted the experiment using various strategies, in both directions (analysis and synthesis kernels, and then averaging the running times) and on three commonly distinct orthogonal wavelet families (Haar, CDF 5/3 and CDF 9/7). Each strategy was tested multiple times (at least 200 times) to measure the average running time. The proposed optimization strategies are summarized as follows:

Using global memory only (*gmem*): In this strategy, global memory was used to hold all buffers and perform the 2D DWT without considering any support of the GPU's on-chip memory (i.e., shared memory, cache, or registers). This approach is a naive GPU implementation, and shows a moderate increase in speed compared to a serial CPU implementation. The results of using only global memory were considered as the baseline to evaluate the efficiency of other optimization techniques.

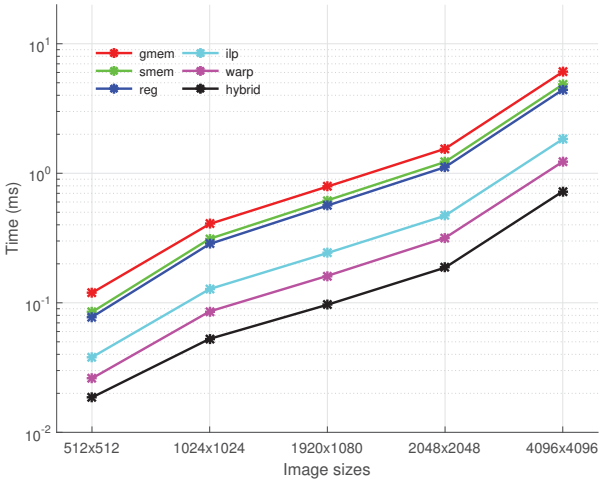
Using shared memory (*smem*): It is essential to move all of the frequently accessed data to a common place, which is shared by a thread block, in order to increase the performance. All the threads within a block copy a tile of the image (plus halo) from global memory and transfer it to shared memory. This helps to reduce the tremendous cost of accessing global memory.

Using registers (*reg*): Instead of invoking `__syncthreads()` after each level of a lifting step in shared memory as above, registers are used for inter-thread communication within a block. This approach allows each GPU thread to work more independently of its neighbors without synchronization.

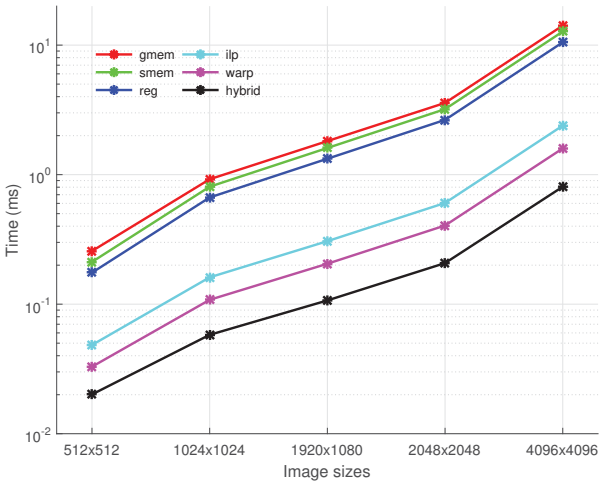
Using instruction level parallelism (*ilp*): The total number of threads is reduced by a factor of 4 and each of them performs more work on either memory operations or wavelet lifting



(a) Haar DWT



(b) CDF 5/3 DWT



(c) CDF 9/7 DWT

Figure 25: Running times (in msecs) of various strategies, on NVIDIA Kepler GPUs.

Table 2: Running times (in msecs) of various optimization strategies for CDF 9/7 DWT on NVIDIA Kepler GPUs

CDF 9/7	*gmem*	*smem*	*reg*	*ilp*	*warp*	*hybrid*
512x512	0.2572	0.2119	0.1751	0.0484	0.0329	0.0201
1024x1024	0.9219	0.8072	0.6667	0.1612	0.1081	0.0580
1920x1080	1.8157	1.6061	1.3269	0.3065	0.2052	0.1069
2048x2048	3.5802	3.2048	2.6476	0.6046	0.4049	0.2076
4096x4096	14.1514	12.7280	10.5169	2.3916	1.5946	0.8066

steps. In this case, the instruction pipeline suffers less from stalls, which leads to higher computational throughput. In addition, it allows thread blocks to leverage more on-chip resources such as registers and shared memory / L1 caches.

Using warp shuffles (*warp*): With the thread block configured as {32, 32} on a {64, 32} tile, the input pixels are first read directly from global memory to registers (declared as an array of 2 by 1 per thread) without `__syncthreads()`. The horizontal pass can proceed with the support of warp shuffles and save the temporary results into shared memory. Then 32 warps can process 64 columns in the vertical pass to complete one level of DWT.

Using hybrid method (*hybrid*): We further extended our strategy to the hybrid parallelism which is a fusion of using registers, warp shuffles, TLP and ILP together. The degree of TLP/ILP was chosen empirically after testing potential configurations in order to have the best choice of representation (see Section 3.2.2).

As shown in Table 2 and illustrated in Figure 25, the running times of various strategies (measured in milliseconds, visualized on a logarithmic scale) are decreasing in the order of approaches (*gmem*, *smem*, *reg*, *ilp*, *warp*, *hybrid*). The results show that those steps we followed leads to a better DWT performance on the GPU.

3.2.2 Comparison with different hybrid configurations

In this section, we show that combining the advantages of TLP (by increasing number of threads) and ILP (by assigning more work per thread) will result in a better DWT performance. As mentioned earlier, the degree of ILP can be adjusted on the basis of the amount of work eligible for one thread, which is the interpolation of *Full-Shuffles* and *Full-Register*. Table 3 specifies the sizes of thread block and the amount of computational work per thread on DWT CDF 9/7,

Table 3: Different block and tile configurations of *hybrid* approach.

TLP \ILP	1	4	8	16	32
32×1	64×1	64×4	64×8	64×16	64×32
(32×4)	–	–	–	64×40	64×104
32×4	64×4	64×8	64×16	64×32	64×64
32×8	64×8	64×16	**64×32**	64×64	64×128
32×16	64×16	64×32	64×64	64×128	64×256
32×32	64×32	64×64	64×128	64×256	64×512

Table 4: Running times (msecs) of *hybrid* CDF 9/7 DWT on a Kepler GPU, image size 1024×1024.

TLP \ILP	1	4	8	16	32
32×1	–	–	–	0.08728	0.23954
32×4	–	–	0.08594	0.06403	0.11589
32×8	–	0.09809	**0.06033**	0.08215	0.14014
32×16	0.14144	0.06940	0.07224	0.12643	–
32×32	0.11320	0.07807	0.10729	–	–

where the position from left to right indicates an increasing degree of ILP (or a decreasing degree of TLP), and the position from top to bottom indicates the thread block size. Note that some of the configurations (in gray) will be invalid due to either the number of necessary pixels along the vertical axis for a valid tile or the hardware limitations (shared memory resource per block). The first row indicates the configuration of the *Full-Register* without the involvement of shared memory. We can assign either one or more warps per blocks to increase the occupancy. The first column covers the *Full-Shuffles* case where 16 warps handle a {64, 16} tile and 32 warps handle a {64, 32} tile. The other configurations belong to the *Semi-Shuffles* where lifting along y is a mixture between intra-thread read and intra-warp data exchange.

We conducted the experiment on various image sizes, e.g., 512×512, 1024×1024 (Table 4), 1920×1080 (Table 5), 2048×2048 and 4096×4096, and observed that the best configuration was using eight warps per block while letting each thread do eight times the amount of computation, i.e., 32×8 block size and 64×32 tile size (marked in boldface).

Table 5: Running times (msecs) of *hybrid* CDF 9/7 DWT on a Kepler GPU, image size 1920×1080.

TLP \ILP	1	4	8	16	32
32×1	–	–	–	0.16205	0.45436
32×4	–	–	0.16094	0.11829	0.21642
32×8	–	0.18526	**0.11145**	0.15333	0.26044
32×16	0.27010	0.12919	0.13411	0.23910	–
32×32	0.21490	0.14499	0.20212	–	–

Table 6: Running times (msecs) of CPU implementations (MATLAB and GSL) and various optimization strategies for CDF 9/7 DWT on a Kepler GPU (including data transfer time).

CDF 9/7	*gsl*	*matlab*	*gmem*	*smem*	*reg*	*ilp*	*warp*	*hybrid*
512x512	3.309	22.600	0.539	0.501	0.491	0.372	0.380	0.356
1024x1024	16.911	91.800	2.222	2.172	2.085	1.589	1.566	1.539
1920x1080	-	167.400	4.632	4.345	4.308	3.325	3.310	3.195
2048x2048	132.794	349.700	9.051	8.846	8.531	6.614	6.416	6.389
4096x4096	635.031	1202.800	35.222	34.090	33.311	25.779	24.961	24.776

3.2.3 Comparison with CPU implementations

In this evaluation, we compare the proposed GPU DWT with several existing CPU DWT implementations to see how much speed up can be achieved from the GPU. The wall-clock running times of each method, which include all the data transfer overhead on GPU implementation for fair comparison, are collected. We chose the DWT implementations available in MATLAB 2015b [8] and GNU Scientific Library (GSL), version 1.16 [48] for this experiment. Table 6 lists the running time of each method measured on various image sizes for one level of CDF 9/7 DWT. As shown in this table, our *hybrid* approach running on a single NVIDIA Kepler GPU K40 (745 MHz) outperforms *gsl* and *matlab* running on an Intel i7-4790K CPU (4.20 GHz) by a large margin – our GPU DWT achieved up to 25× and 65× speed up, respectively.

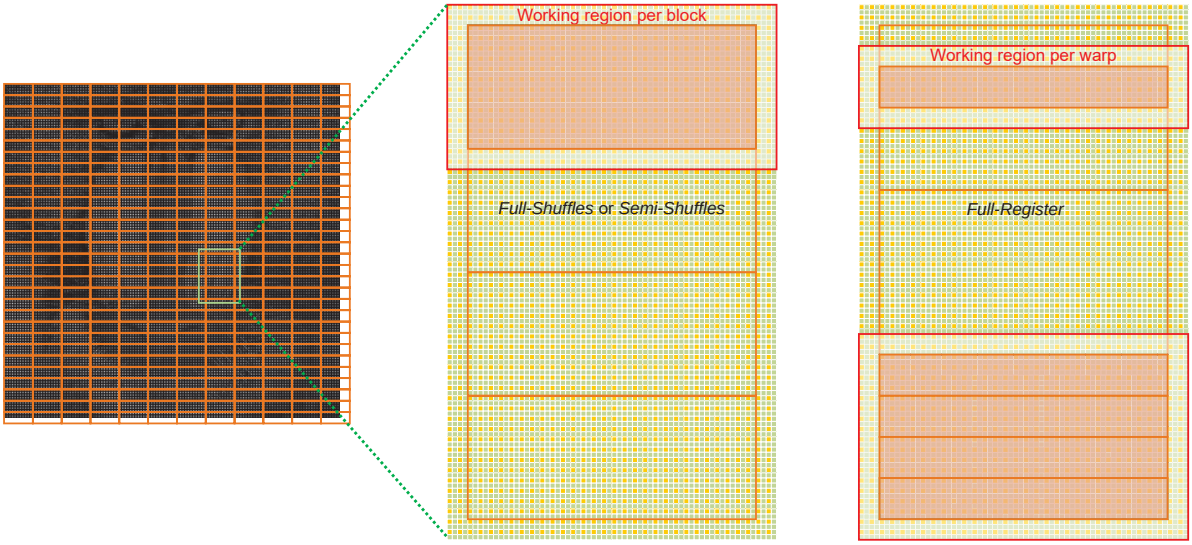


Figure 26: Overlapping regions of *Full-Shuffles* or *Semi-Shuffles*, compared to *Full-Register*.

Table 7: Running times (in msecs) of other GPU DWT methods and the proposed methods for CDF 9/7 DWT.

CDF 9/7	GF100 (Fermi)						GK110 (Kepler)	
	[92]	[46]	[115]	[125]	[114]	*ilp*	[40]	*hybrid*
512 × 512	0.51	0.35	0.46	0.25	0.16	0.107	0.0545	0.0204
1024 × 1024	1.31	1.08	1.05	0.66	0.46	0.387	0.1452	0.0594
1920 × 1080	2.36	2.14	1.79	1.21	0.86	0.747	–	0.1114
2048 × 2048	4.51	3.86	2.54	1.98	1.68	1.489	0.4812	0.2227
4096 × 4096	17.25	16.38	14.15	7.18	6.44	5.924	1.7919	0.8678

3.2.4 Comparison with other GPU DWT methods

We compared our method with the most recent GPU wavelet work using Fermi GPUs [114] and its references therein. In other to make direct comparisons, we ran our CDF 9/7 DWT on an NVIDIA GF100 GPU (Tesla C2050) and compared our results with the results of methods from [114].

To be more specific, Matela [92] and our early work [100] used a symmetric block-based approach, which disregards the halo across a block boundary, and hence acts as an approximation of the classical filter scheme. The other methods, including our proposed method, are an exact

[8] `http://www.mathworks.com/products/wavelet/`

implementation of non-Haar DWT. Among those, Franco et al. [45], Song et al. [115], and Laan et al. [126] split the horizontal and vertical pass into two cascaded stages, which differs in the handling of the intermediate result before performing the vertical transform. Franco et al. [45] included the transposition after each pass so that on the next stage of wavelet transform, data would already be laid out the same as on the horizontal pass. Song et al. [115] did not have the transposition step, but instead processed the vertical pass in column segment fashion. As showed in Table 7, Franco et al. [45] was faster than Song et al. [115] only on small image (512×512) where the data was actively cached on the transpose operation. It suffered from the memory bottleneck, however, due to the data movement for larger images. Laan et al. [126] proposed a slab approach (sliding window) to achieve a fast column transform where enough data would be stored in the top and the bottom of the slab, and multi-column could be processed at the same time to maximize the use of shared memory. Overall, those three methods involved expensive global memory in-between and that impaired their performance.

The recent improvement from Song et al. [114] employed the block-based (or tiling) technique, which is same as ours, to eliminate the intermediate results being stored in global memory. This approach performs a two-pass transform in one kernel call, and hence establishes a 2D stencil-based processing procedure. Song et al. also implicitly used the ILP where the block was configured as {60, 1} and the tile was set as {60, 32}. Equivalently, they assigned 2 warps (64 threads) for one tile of the image and hence 4 threads out of 64 were idling. Note that this approach did not fully leverage the GPU resource when the block size was not divisible by the warp size (32), which left more room to optimize further.

Our *ilp* approach is faster (Table 7) than the recent method from Song et al. [114] because we employed ILP and used fast bitwise computation to calculate the subband's locations. It also showed that the advantages of giving more work per thread will lead to better performances. Because we just need to store the valid coefficients (neglect the halo) to global memory, there are still idling threads during the last pass of writing (see Figure 23). The ratio between the active and the total threads of our method is 88.5%, however, which is much higher than 79.4% from Song et al. [114].

On NVIDIA Kepler architecture

We also compared our method with the most recent register-based GPU DWT algorithm [40]. In order to make a direct comparison, we downloaded their source code from the web repository (available at [9]) and compared it with our *hybrid* version on an NVIDIA K40 GPU. Again, one

[9] `http://github.com/PabloEnfedaque/`

level of CDF 9/7 DWT was used as a test case to stress the computing power of the GPU. As shown in Table 7, our *hybrid* method consistently outperformed Enfedaque et al. [40].

The main reason behind that result is that once ILP is saturated and the maximum performance is reached, then assigning more work to a thread cannot increase the performance. Since ILP and TLP are inversely proportional to each other for the fixed problem size, if ILP is increased then TLP is decreased, and vice versa. Just maximizing ILP would hurt TLP significantly. In [127], Volkov et al. showed that maximizing ILP can result in an optimal performance even though occupancy becomes very low, i.e., the degree of TLP is low. Along this line, Enfedaque et al. [40] introduced a GPU DWT that maximized ILP using registers only. As shown in the recent study by Fatehi and Gratz [44], however, there exists an upper bound of ILP for total memory and computation instructions. Therefore, we believe that combining TLP with ILP can be more effective because memory and instruction latency can be further hidden by concurrent threads once ILP reaches its peak. We also believe that there should be enough concurrent warps running on a Streaming Multiprocessor (SMX) of the GPU for optimal performance. For example, on NVIDIA Kepler architecture, up to two independent instructions from each of the four concurrent warps can be issued per clock cycle [10], which makes the setup of eight warps per block in the *hybrid* optimization fit better to Kepler GPUs than did the four warps per block in Enfedaque et al [40].

Another important point is that our *hybrid* method used both warp shuffles and registers for vertical (along y axis) lifting steps, which allowed fewer overlapping regions than Enfedaque et al [40]. For example, consider the case of performing CDF 9/7 DWT on an image: its {64, 96} portion must use four blocks (each has 8 warps) to proceed (as *Full-Shuffles* or *Semi-Shuffles*). This results in a ratio of the overlapping halo region to the output size of 45.83%. By comparison, *Full-Registers* requires three blocks (each has 4 warps) and hence, has 112.5% of the overlapping ratio. This is mainly because the working area of *Full-Shuffles* or *Semi-Shuffles* is per-block while *Full-Registers* or Enfedaque et al. [40] is per-warp.

3.2.5 Comparison on multi-level GPU DWT

In this section, we assess the actual application-level running time of the proposed GPU DWT for multi-level transformation to see how the proposed method performs under realistic conditions. Table 8 shows the running times of CUDA kernels measured using a wall-clock timer for up to four levels of CDF 9/7 DWT in our *hybrid* approach and Enfedaque et al. [40]. We used test images at various resolutions ranging from 512×512 to 4096×4096. For accuracy, we ran the same

[10] `http://docs.nvidia.com/cuda/kepler-tuning-guide/`

Table 8: Running times (in msecs) of [40] and *hybrid* for multi-level CDF 9/7 DWT.

CDF 9/7	Enfedaque et al. [40]				*hybrid*			
level	1	2	3	4	1	2	3	4
512x512	0.0545	0.0835	0.1107	0.1364	0.0204	0.0407	0.0544	0.0670
1024x1024	0.1452	0.2006	0.2259	0.2533	0.0594	0.1015	0.1197	0.1334
2048x2048	0.4812	0.6251	0.6770	0.7064	0.2227	0.3505	0.3878	0.4057
4096x4096	1.7919	2.2756	2.4170	2.4733	0.8678	1.3525	1.4717	1.5086

Table 9: Running times (in msecs) of *hybrid* and *mb-hybrid* for multi-level Haar DWT, on an NVIDIA Kepler GPU.

Haar	*hybrid*				*mb-hybrid* (fused kernel)			
level	1	2	3	4	1	2	3	4
512x512	0.0150	0.0298	0.0414	0.0524	0.0129	0.0152	0.0182	0.0192
1024x1024	0.0458	0.0753	0.0901	0.1016	0.0333	0.0417	0.0522	0.0560
2048x2048	0.1810	0.2810	0.3106	0.3251	0.1139	0.1472	0.1888	0.2039
4096x4096	0.7162	1.1165	1.2181	1.2479	0.4470	0.5690	0.7336	0.7934

test multiple times and collected the average running time. Each test includes running times for both forward and inverse transformations. As shown here, our proposed method achieved higher performance compared to the state-of-the-art method [40] measured on an NVIDIA Kepler GPU (K40). We observed that our *hybrid* method runs up to 2.6× faster than Enfedaque et al., and its performance gap is slowly reduced as the number of transform levels and image size increase.

Table 9 shows the running times of our previous hybrid optimization on the conventional memory layout (*hybrid*) and the mixed-band layout (*mb-hybrid*) for fused multi-level Haar DWT. In the *hybrid* approach, the transformation result at every level is written to global memory, but the *mb-hybrid* approach transforms the input image up to four levels without accessing global memory. This results in a significant performance improvement, up to 2.7× speed up over the *hybrid* approach that is the most optimized version for single level wavelet transformation.

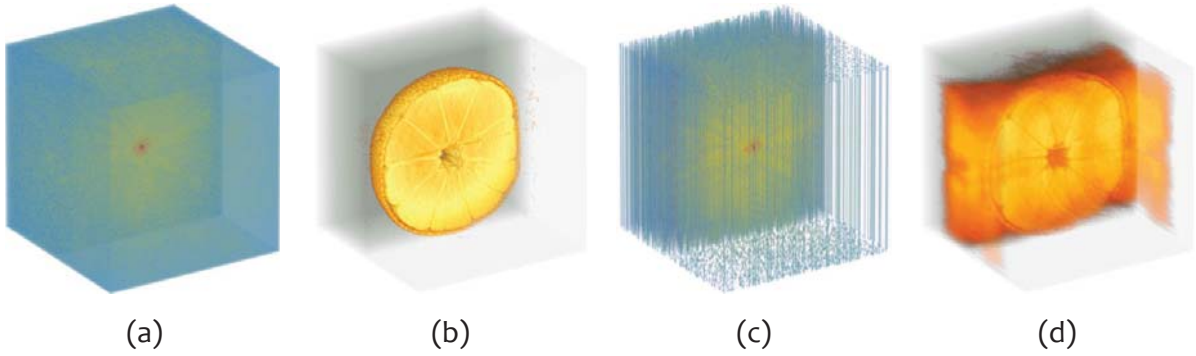


Figure 27: Full 3D k-space (a), ×4 undersampling k-space (c) and their reconstructions (b), (d).

3.3 Application: 3D CSMRI reconstruction

To further assess the usability of the proposed GPU mixed-band lifting wavelet transform, we applied our wavelet transform as a regularizer in a 3D CSMRI reconstruction problem, which can be generally formulated as follows:

$$\min_{s} \{J(s)\} \quad s.t. \quad \sum \|RFs - m\|_2^2 < \sigma^2 \tag{10}$$

where $J(s)$ stands for the energy function we want to minimize, f is the measurement of the MRI data (i.e., *k-space*), and $K = RF$ is the matrix multiplication that combines a sampling pattern R and a Fourier matrix F for 3D data. Examples of k-spaces and their reconstructions are shown in Figure 27. $J(s)$ can comprise many terms, for example Total Variation (TV) [51], Wavelet constraint [51, 86], Fourier constraint [71, 86], and the fidelity of the data. More recent GPU implementations on MRI reconstructions with different solvers can be found in [94] and references therein.

For instance, a 3D CSMRI energy function is described as follows:

$$J(s) = \|\nabla_{xyz} s\|_1 + \|W_{xyz} s\|_1 \tag{11}$$

In this equation, ∇ is the gradient operator that enforces the smooth variation of pixel intensity in spatial dimensions, and W is a 3D DWT. In order to solve the minimization problem of Eq. 12 using the energy function given above, we use the Split Bregman iterative algorithm [51].

We include the wavelet term in $J(s)$ and compare its reconstruction to the zero filling version and without wavelet, on 3D fruit MRI in term of Peak-Signal-To-Noise ratio (PSNR) to the full reconstruction. The experiments are conducted with ×4 and ×8 mask patterns on the kiwi and tangerine data set (size 128×128×128). It took ∼7 seconds to run the solver on the GPU with the number of inner and outer loops are 32 and 16, respectively. Kim et al. [74] reported a

Table 10: PSNRs of 3D CSMRI reconstruction on 128×128×128 datasets. Unit: dB

Dataset	Mask	Zero filling	No Wavelet	With Wavelet
Kiwi	×4	18.7245	27.8483	28.3488
	×8	17.2263	25.6509	26.1510
Tangerine	×4	22.0418	35.6646	43.6321
	×8	20.6875	33.3996	36.2106

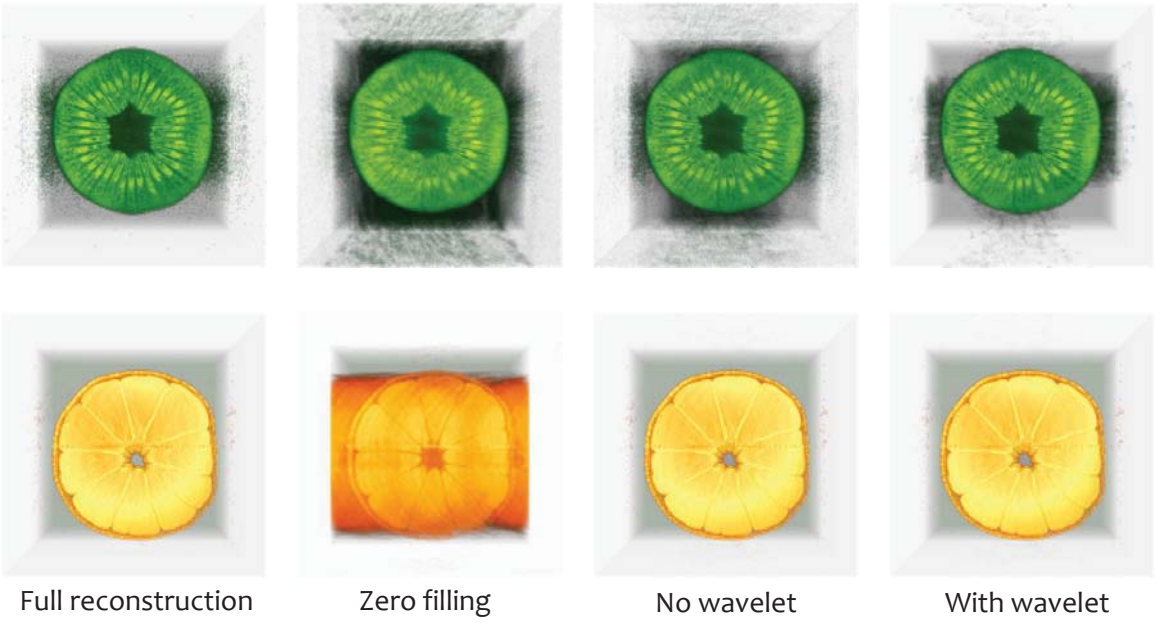


Figure 28: CSMRI reconstruction on 3D data with ×4 mask with kiwi and tangerine datasets

similar study result (3D Split Bregman on the GPU) in [74]. Although it is difficult to make a direct comparison between two methods since the testing environment and energy function are different, we concluded that our method is roughly ∼2.5× faster than their method after proper scaling of parameters. Note that Kim et al. did not include wavelet transform so our CSMRI solver is heavier in arithmetic intensity, but still outperforms Kim's method for the same number of iterations.

As shown in Table 10, adding a 3D wavelet term can effectively increase the PSNR of the reconstructed images. Figure 28 visualizes the CSMRI 3D volumes of kiwi and tangerine data which have been cut at the middle to show the inside. It can be seen that the wavelet term helps to decrease the noise-like effects and enhance the quality of 3D MRI reconstruction.

3.4 Summary

In this work, we introduced various optimization strategies for 2D discrete wavelet transforms on the GPU. The proposed strategies leverage fast on-chip memories (shared memory and registers), warp shuffle instructions, and thread- and instruction-level parallelism. We showed that, unlike other state-of-the-art GPU DWTs, hybrid parallelism that exploits both ILP and TLP together results in the most optimal performance. We also showed that the mixed-band layout of Haar DWT outperformed the conventional DWT especially when multi-level transformation is taken into account. For future work, we plan to apply our proposed GPU DWT to various wavelet applications, e.g., compressed sensing MRI reconstruction and sparse coding using dictionary learning, and investigate the performance benefits in large-scale data processing applications.

IV Multi-GPU Reconstruction of dCS-MRI

4.1 Method

The input to our system is 2D DCE MRI data (a collection of 2D k-space data over time) defined on a 3D domain by the parameter (x, y, t), where x and y are spatial coordinates for 2D k-space data, and t denotes a temporal coordinate on the time axis. In the following, the subscript of an operator represents its dimension, e.g., F_{xy} stands for a Fourier operator applied to a 2D x-y grid.

4.1.1 Compressed Sensing formulation for DCE-MRI

Let S be a collection of 2D MRI k-space data s_i, i.e., $S = \{s_1, s_2, ..., s_n\}$, acquired over time. Then the general CS formulation for DCE-MRI reconstruction problem can be described as follows:

$$\min_S \{J(S)\} \quad s.t. \quad \sum_i \|Ks_i - m_i\|_2^2 < \varepsilon \tag{12}$$

where $J(S)$ is the ℓ_p-norm energy function to minimize (i.e., regularizer), f_i is the measurement at time i from the MRI scanner, and $K = RF$ is the sampling matrix that consists of a sampling mask R and a Fourier matrix F for 2D data. Since DCE-MRI does not change abruptly along the time axis, we enforce the temporal coherency by introducing a total variation energy along t axis and decouple the sparsifying transform on the x-y plane and temporal axis as follows:

$$J(S) = \|W_{xy}S\|_1 + \|\nabla_{xy}S\|_1 + \|\nabla_t S\|_1 \tag{13}$$

where W is the wavelet transform. Then Equation 12 can be solved as a constrained optimization problem using a Bregman iteration [51] where each update of S^{k+1} in line 3 solves an unconstrained problem as shown in Algorithm 1.

Algorithm 1 Constrained CS Optimization Algorithm

1: $k = 0, s_i^0 = m_i^0 = \mathbf{0}$ for all i
2: **while** $\sum_i \left\|R_i F_{xy} s_i^k - m_i\right\|_2^2 > \varepsilon$ **do**
3: $\quad S^{k+1} = \min_S \{J(S) + \frac{\mu}{2} \sum_i \left\|R_i F_{xy} s_i - m_i^k\right\|_2^2\}$
4: $\quad m_i^{k+1} = m_i^k + m_i - R_i F_{xy} s_i^{k+1}$ for all i
5: **end while**

Using the Split Bregman algorithm [51], we can decouple ℓ_1 and ℓ_2 components in line 3 in Algorithm 1 and iteratively update using a two-step process (more details can be found in [51]). Note that in the original Split-Bregman algorithm, S and $J(S)$ are defined on the same

dimensional grid, i.e., if S is a 3D volume then $J(S)$ is a hybrid of 3D total variation (TV) and wavelet regularizers, which allows a closed-form solution for S^{k+1}. However, because our $J(S)$ consists of 1D and 2D operators, the same closed-form solution is not applicable. To be more specific, S^{k+1} can be updated by solving the linear system defined as follows:

$$(\mu F_{xy}^{-1} R^T R F_{xy} - \lambda \Delta_{xy} - \theta \Delta_t + \omega) S^{k+1} = rhs^k \tag{14}$$

where the parameters λ, θ, ω and μ are used to control the amount of regularization energy, and refer to [51] for the definition of rhs^k. Goldstein et al. [51] proposed a closed form solution to invert the left-hand side of the given linear system using the forward and inverse 3D Fourier transforms, but we cannot apply the same method since the k-space data is 2D in our formulation.

Since the left-hand side of Equation 14 is not directly invertible, we can use iterative methods, such as the fixed-point iteration or the conjugate gradient method, that converge slower than the explicit inversion in the original method. In order to speed up the convergence, we propose a *single iteration* method, which further reduces the iteration cost by splitting the left-hand side of Equation 14 and solve for S^{k+1} directly in a single step. If we separate 1D and 2D operators in the left-hand side, then Equation 14 can be represented as follows:

$$(A_1 + A_2) S^{k+1} = rhs^k \tag{15}$$

where $A_1 = (\mu F_{xy}^{-1} R^T R F_{xy} - \lambda \Delta_{xy})$ and $A_2 = -\theta \Delta_t + \omega$. If we treat A_2 as the operator applied to S from the previous iteration, i.e., S^k at $(k+1)$-th iteration, then Equation 15 can be expressed as follows:

$$A_1 S^{k+1} + A_2 S^k = rhs^k \tag{16}$$

Because S^k is known, we can move it to the right-hand side to make the update rule for S^{k+1}

$$S^{k+1} = A_1^{-1}(rhs^k - A_2 S^k) \tag{17}$$

This assumption holds for a sufficiently large k because S^k and S^{k+1} will converge. Then, A_1 can be inverted by making the system circulant as shown in [51]

$$S^{k+1} = F_{xy}^{-1} \mathcal{K}^{-1} F_{xy} (rhs^k + (\theta \Delta_t - \omega) S^k) \tag{18}$$

where $\mathcal{K}$ is the diagonal operator defined as $\mathcal{K} = \mu R^T R - \lambda F_{xy} \Delta_{xy} F_{xy}^{-1}$.

We evaluated the convergence rates of three different minimization strategies for S^{k+1} – fixed-point iteration, conjugate gradient, and single iteration methods (Fig. 29). We observed that the conjugate gradient method converges about twice faster than the fixed-point iteration, and our single iteration converges about 2∼3-fold faster than the conjugate gradient method to reach the same PSNR.

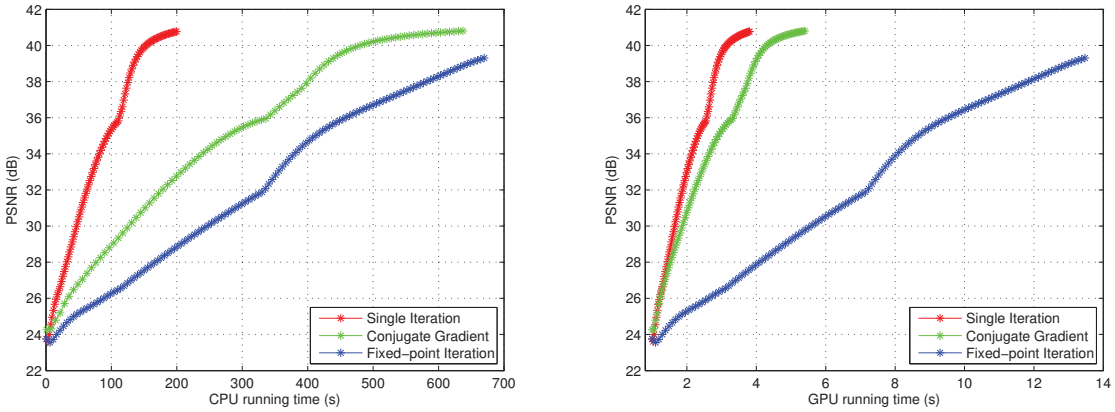


Figure 29: PSNR vs. CPU (left) and GPU running times (right) of various methods.

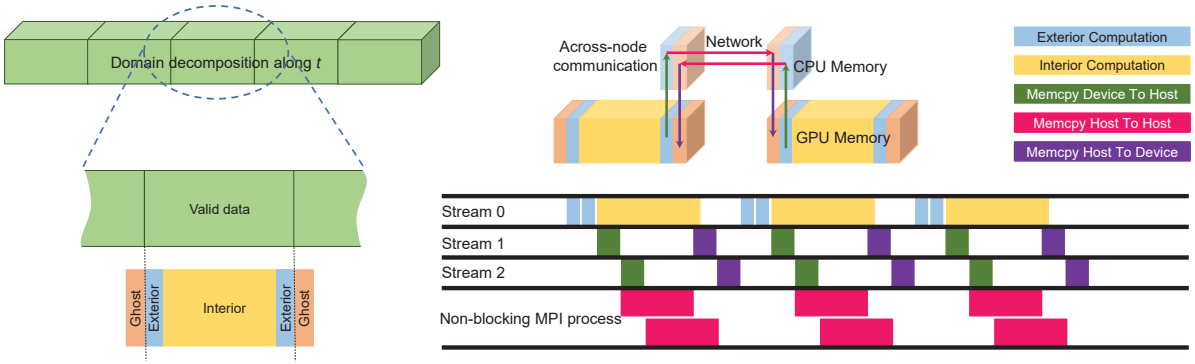


Figure 30: Ghost region communication between neighbour GPUs

4.1.2 Multi-GPU Implementation

As the temporal resolution of DCE-MRI increases, the entire k-space can not be mapped onto a single GPU. We decompose the domain along the t-axis into chunks where each chunk consists of three parts: *interior*, *exterior*, and *ghost*, similar to [93]. Interior and exterior regions form a computational domain for a given chunk, and ghost is the region extends to its neighbor chunk that is required for computation (Fig. 30 left). In our method, we increase the ghost and exterior size (up to 5 slices in our experiment) in order to run the algorithm multiple iterations without communication.

Data communication between adjacent GPUs is performed as follows (see Fig. 30 bottom right): First, we perform the computation on the exterior regions, which are the ghost regions of the neighborhood GPU, using the main stream (stream 0, Fig. 30 cyan). Note that we can run n iterations in this step for the ghost size n (one slice of ghost region is invalidated per each iteration, so we can run up to n iterations). Next, the other concurrent streams (stream 1 and

2) will perform the peer copies from GPU (device) to CPU (host) memory, one stream per an exterior region (Fig. 30 green). Because streams run in parallel, stream 0 can continue to compute on the interior region during the communication (Fig. 30 yellow). Then each exterior region data on the CPU will be transferred to the ghost region of its neighborhood CPU via MPI send and receive (Fig. 30 magenta). Because the MPI process becomes available right after invoking the GPU asynchronous calls, it can be used to perform a non-blocking transaction across the GPUs (note that stream 2 is overlapped with MPI transfer). Finally, the valid ghost data is copied from the host to device asynchronously to complete one round of ghost communication (Fig. 30 purple). By doing this, we can effectively hide the communication latency by allowing the ghost transfers occur on the different GPU streams while the main stream continues the computation on the interior region. If GPUs are in the same physical node, this communication can be implemented using asynchronous peer-to-peer device communication via PCI bus. If the system supports infiniband network, we can use NVIDIA GPUDirect for RDMA communication.

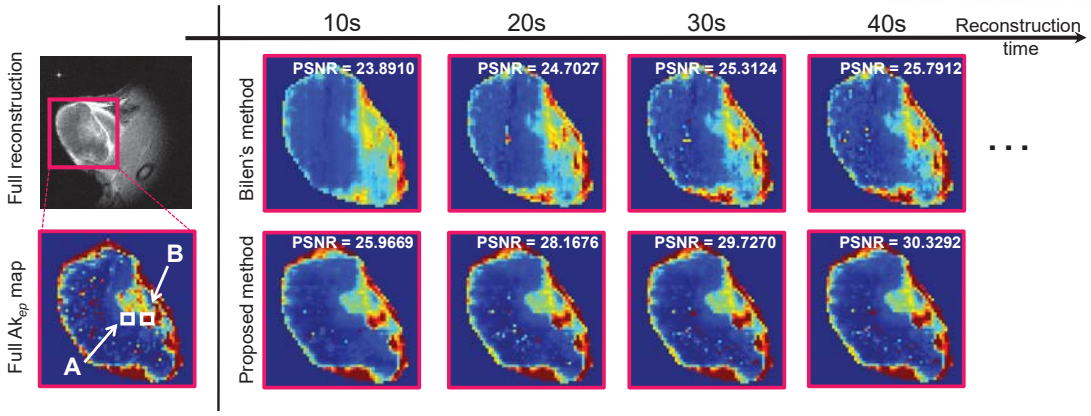
4.2 Result

We used an NVIDIA GeForce GTX 680 GPU with Jacket for MATLAB for the single GPU evaluation (same environment as Bilen's). We ran our scalability test on a 6-node GPU cluster system with two NVIDIA K20m per node, 12 GPUs in total. Gd-DTPA contrast agent was used for data preparation, and the CS-undersampling factor is $\times 8$.

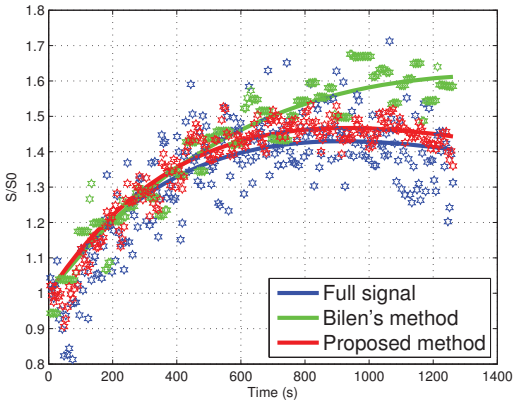
4.2.1 Image Quality and Running time Evaluation

We compared our methods with IMPATIENT [47], kt-SPARSE [86], kt-FOCUSS [71], GPU Split-Bregman [74] and GPU ADMM [9], and our method is comparable to or outperforms those. Among them, we discuss Bilen et al. [9] in detail because it is the most recent work close to our method in the sampling strategy, energy function, numerical method, and GPU implementation. For a direct comparison with Bilen's, the wavelet term in the formulation is ignored in this experiment.

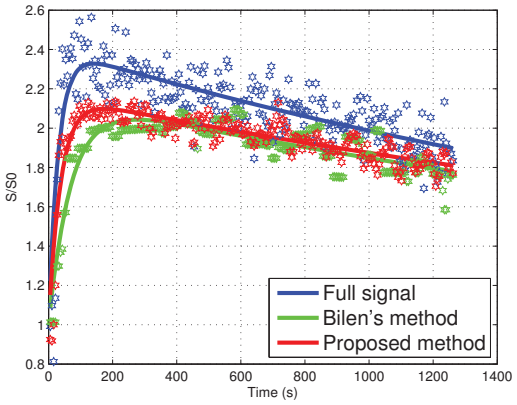
Figure 31 (a) compares Ak_{ep} maps [63] of reconstructed images at different points in time. The Ak_{ep} maps in the bottom row (our method) is much closer to the ground truth (Full Ak_{ep} on the left) than those in the upper row at the same point in time, which implies our method converges faster. Figure 31 (b) and (c) show the least-square fitting curve to the temporal profile of the region of interest in the Ak_{ep} map (A and B, high vascularity regions), which is also a commonly used quality measure to characterize the perfusion and permeability in DCE-MRI data. For this test, we ran each method until its reconstruction image reaches the same PSNR.



(a) Ak$_{ep}$ maps at specific moments of reconstruction time



(b) Temporal profile of A



(c) Temporal profile of B

Figure 31: Comparison between Bilen's method and the proposed solution

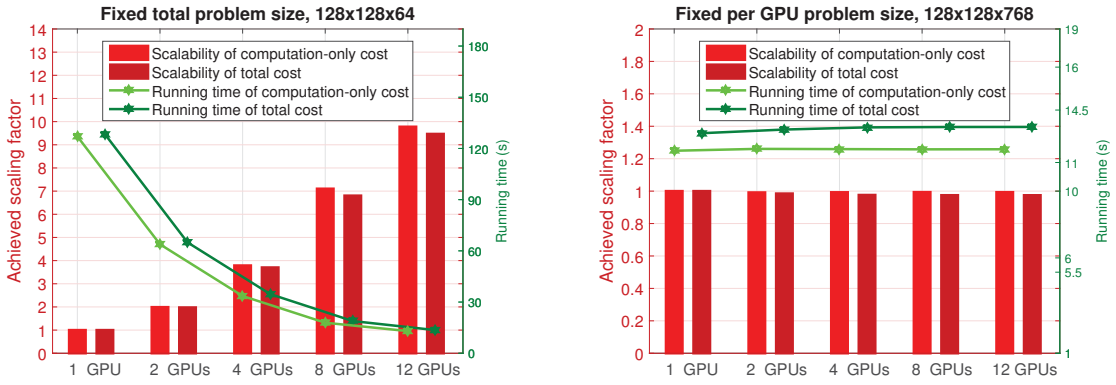
While both methods generate the curves similar to that of the full reconstruction, our result is more accurate (i.e., close to the full reconstruction curve), which is due to the different numerical characteristics of the algorithm. Table 11 shows the running time of each method on several tumor datasets until convergence (i.e., reaching a steady state). The result also confirms that our method converges much faster than Bilen's, up to 6.9× in GPU implementation.

4.2.2 Multi-GPU Performance Evaluation

In this evaluation, we check the scalability up to 12 GPUs on a distributed GPU cluster. We first measure the computation-only time to obtain the best possible running times, and then measure the total running times including data communication (i.e., ghost region exchange). As shown in Figure 32, the total time including the ghost exchange is approximately close to the computation-only time, in both strong and weak scaling tests. This result confirms that our multi-GPU implementation can effectively hide the communication latency while performing the CS DCE-MRI reconstruction solver on distributed systems.

Table 11: Running times of Bilen's and the proposed method on datasets of size $128 \times 128 \times 256$.

Metrics	Tumor 1		Tumor 2		Tumor 3		Tumor 4	
	Bilen	Ours	Bilen	Ours	Bilen	Ours	Bilen	Ours
PSNR(dB)	30.598	30.636	40.169	39.931	39.678	39.479	33.559	34.174
CPU time(s)	448.451	360.689	797.056	340.143	719.187	327.868	715.004	407.461
GPU time(s)	292.341	61.637	475.155	68.002	394.241	66.125	367.759	63.276



(a) (b)

Figure 32: Strong (a) and weak scaling (b) on a distributed GPU cluster.

4.3 Summary

In this work, we presented our new CS-based DCE-MRI reconstruction system for the multi-GPU computing environment. The proposed method delivered a new numerical method in order to apply the Split-Bregman algorithm to CS-based time-variant DCE-MRI problem. We also introduced a scalable implementation of the proposed CS-MRI reconstruction method on a distributed multi-GPU system. As discussed, the proposed method outperforms the existing GPU CS-reconstruction algorithm in quality and running time. For future work, we plan to extend the proposed CS-MRI method to large-scale dynamic 3D DCE-MRI reconstruction. Assessing the clinical feasibility of the proposed method would be another interesting future research.

V 2D Convolutional Sparse Coding Reconstruction of dCS-MRI

In this part, we introduce a novel application of convolutional sparse coding to reconstruct undersampled DCE-MRI. We leverage the property of DCE-MRI that many similar local features may appear over time due to its temporal coherency so that learning a dictionary from the images over the range of time can improve the reconstruction quality better than using the temporal Total Variation energy only. In contrast to the conventional dictionary learning-based CS-MRI [17], our method can avoid expensive patch-based learning by embedding the convolution operator directly into the energy function and its solution can be efficiently computed in the frequency domain, i.e., k-space, using an alternating method. We also show that the proposed method maps well to data-parallel architecture, such as GPUs, for further accelerating its running time significantly. To the best of our knowledge, this is the first CS-DCE-MRI reconstruction method based on convolutional sparse coding and GPU acceleration.

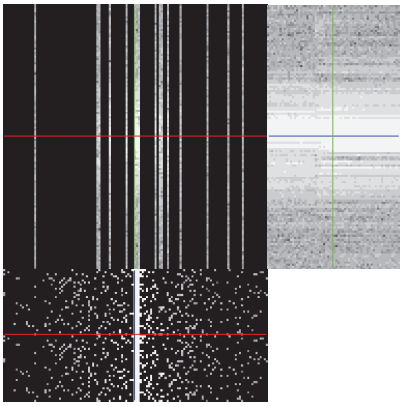
5.1 Method

Figure 33 is a pictorial description of the proposed method. If the inverse Fourier transform is directly applied to undersampled DCE-MRI k-space data (Fig. 33a, $\times 8$ undersampling), the reconstructed images will suffer from artifacts (Fig. 33b). The zero-filling reconstruction shown above will serve as an initial guess (Fig. 33c) for our iterative reconstruction process with randomly initialized filters, e.g., a collection of 100 atoms of size 21×21 as shown in Fig. 33d. Then the image and filters are iteratively updated until they converge as shown in Fig. 33e and f. The proposed CS-DCE-MRI reconstruction algorithm is a process of finding S_i (i.e., DCE MR image at the time step i) of the energy minimization problem defined as follows:

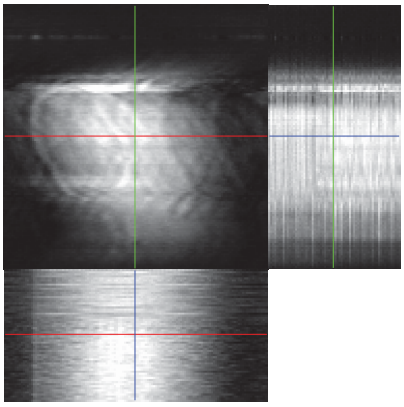
$$\min_{D_k, X_{k,i}, S_i} \frac{\alpha}{2} \left\| \sum_k D_k * X_{k,i} - S_i \right\|_2^2 + \lambda \sum_k \|X_{k,i}\|_1 + \theta \|\nabla_t S_i\|_1$$
$$s.t. \quad \|R_i F S_i - M_i\|_2^2 < \varepsilon, \|D_k\|_2^2 \leq 1 \tag{19}$$

where D_k is the k-th filter (or atom in the dictionary) and $X_{k,i}$ is its corresponding sparse code for S_i. In order to reconstruct the entire DCE MR images for t time steps, we should solve Equation (19) for $i = 0, ..., t-1$.

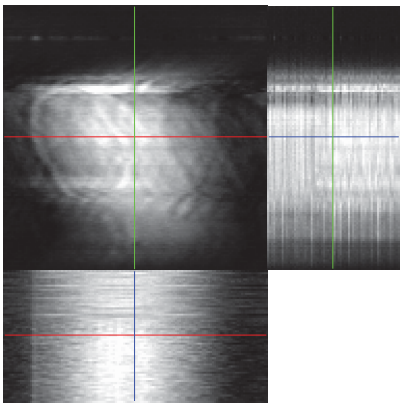
In Equation (19), the first term measures the difference between S_i and its sparse approximation $\sum_k D_k * X_{k,i} - S_i$ weighted by α. The second term is the sparsity regularization of $X_{k,i}$ using an ℓ_1 norm with a weight λ instead of an ℓ_0 norm as used in [1, 17]. The third term $\theta \|\nabla_t S_i\|_1$ is the Total Variation energy that enforces the temporal coherence of DCE data, which is widely used in conventional CS-DCE-MRI reconstruction algorithms [86, 99]. The rest of the equation is the collection of constraints – the first constraint enforces the consistency



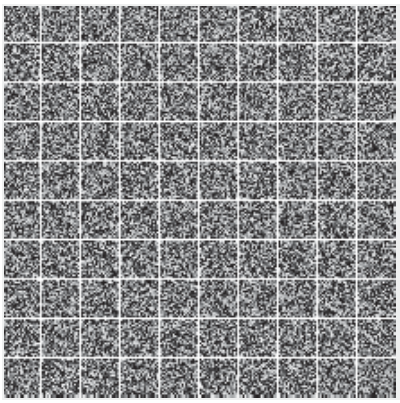
(a) Undersample k-space



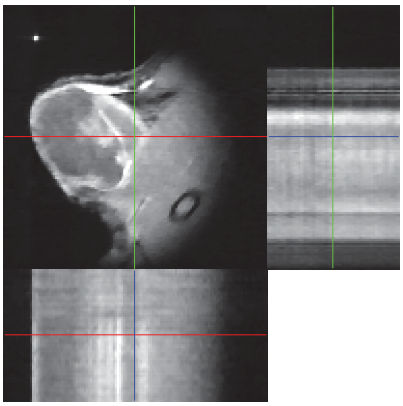
(b) Zero-filling reconstruction



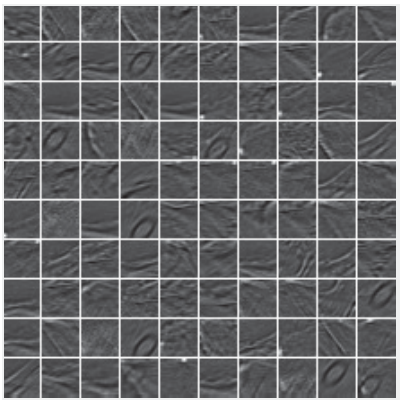
(c) Initial guess



(d) Ramdomly initialized filters



(e) Converged solution



(f) Converged filters

Figure 33: CS-DCE-MRI Reconstruction using convolutional sparse coding. Red line: t axis, Blue–green lines: x∼y axis

between each undersampled measurement M_i and the undersampled reconstructed image using the mask R_i and the Fourier operator F, and the second constraint restricts the Frobenius norm of each atom D_k within a unit length. In the following sections, we split Equation (19) into two sub optimization problems and alternate between them to find the global minimum solution for $\mathbf{S} = \{S_0, S_1, ..., S_{t-1}\}$. In the following discussion, we will use a simplified notation without indices k and i and replace the result of Fourier transform of a given variable by using the subscript f to derive the solution of Equation (19).

5.1.1 Subproblem for convolutional sparse coding

The first sub optimization problem of Equation (19) is the formulation of convolutional sparse coding as follows:

$$\begin{aligned} \min_{D,X,S} \quad & \frac{\alpha}{2}\left\|\sum D * X - S\right\|_2^2 + \lambda \sum \|X\|_1 \\ & s.t.\ \|RS_f - M\|_2^2 < \varepsilon,\ \ \|D\|_2^2 \leq 1 \end{aligned} \tag{20}$$

Problem (20) can be rewritten using auxiliary variables Y and G for X and D:

$$\begin{aligned} \min_{D,G,X,Y,S} \quad & \frac{\alpha}{2}\left\|\sum D * X - S\right\|_2^2 + \lambda \sum \|Y\|_1 \\ s.t.\ & X = Y,\ \|RS_f - M\|_2^2 < \varepsilon,\ G = \mathcal{P}roj(D),\ \ \|G\|_2^2 \leq 1 \end{aligned} \tag{21}$$

where G and D are related by a projection operator as a combination of a truncated matrix followed by a padding-zero matrix in order to make the dimension of G same as that of X. Since we will leverage Fourier transform to solve this problem, G should be padded to zero to make its size same as G_f and X_f. The above constraint problem can be rebuilt in an augmented Lagrangian form with dual variables U, H, and further regulates the measurement consistency and the dual differences with γ, ρ, and σ, respectively:

$$\begin{aligned} \min_{D,G,X,Y,S} \frac{\alpha}{2}\left\|\sum D * X - S\right\|_2^2 + \frac{\gamma}{2}\|RS_f - M\|_2^2 + \lambda \sum \|Y\|_1 + \frac{\rho}{2}\|X - Y + U\|_2^2 + \frac{\sigma}{2}\sum \|D - G + H\|_2^2 \\ s.t. \quad G = \mathcal{P}roj(D), \quad \|G\|_2^2 \leq 1 \end{aligned} \tag{22}$$

Then we can solve problem (22) by iteratively finding the solution of independent smaller problems, as described below:

Solve for X:

$$\min_X \frac{\alpha}{2}\left\|\sum D * X - S\right\|_2^2 + \frac{\rho}{2}\|X - Y + U\|_2^2 \tag{23}$$

If we apply the Fourier transform to the (23), it becomes:

$$\min_{X_f} \frac{\alpha}{2}\left\|\sum D_f X_f - S_f\right\|_2^2 + \frac{\rho}{2}\|X_f - Y_f + U_f\|_2^2 \tag{24}$$

Then the minimum solution of (24) can be found by taking the derivative of (24) with respect to X_f and setting it to zero as follows:

$$\left(\alpha \sum D_f^H D_f + \rho I\right) X_f = \alpha \sum D_f^H S_f + \rho \left(Y_f - U_f\right) \tag{25}$$

Solve for Y:

$$\min_Y \quad \lambda \sum \|Y\|_1 + \frac{\rho}{2} \|X - Y + U\|_2^2 \tag{26}$$

Y for ℓ_1 minimization problem can be found by using a shrinkage operation:

$$Y = \mathcal{S}hrink_{\lambda/\rho} \left(X + U\right) \tag{27}$$

Solve for U: The update rule for U can be defined as a fixed-point iteration with the difference between X and Y (U converges when X and Y converge each other) as follows:

$$U = U + (X - Y) \tag{28}$$

Solve for D: Similar to (25) , D can be solved in the Fourier domain:

$$\left(\alpha \sum X_f^H X_f + \sigma I\right) D_f = \alpha \sum X_f^H S_f + \sigma \left(G_f - H_f\right) \tag{29}$$

Solve for G: G can be found by taking the inverse Fourier transform of D_f. This projection should be constrained by suppressing the elements which are outside the filter size D_k, and followed by normalizing its ℓ_2-norm to a unit length.

$$\min_G \ \frac{\sigma}{2} \|D - G + H\|_2^2 \quad s.t \ \ G = \mathcal{P}roj(D), \ \ \|G\|_2^2 \leq 1 \tag{30}$$

Solve for H: Similar to U, the update rule for H can be defined as follows:

$$H = H + (D - G) \tag{31}$$

Solve for S:

$$\min_S \ \frac{\alpha}{2} \|D * X - S\|_2^2 + \frac{\gamma}{2} \|RS_f - M\|_2^2 \ \ s.t. \ S_f = F(S) \tag{32}$$

Similar to (24), the objective function of (32) can be transformed into Fourier domain:

$$\min_{S_f} \quad \frac{\alpha}{2} \left\| \sum D_f X_f - S_f \right\|_2^2 + \frac{\gamma}{2} \|RS_f - M\|_2^2 \tag{33}$$

Then S_f can be found by solving the following linear system:

$$\left(\gamma R^H R + \alpha I\right) S_f = \gamma R^H M + \alpha \sum D_f X_f \tag{34}$$

Note that the efficient solutions of (25), (29) and (34) can be obtained via the Sherman-Morrison formula for independent linear systems as shown in [131].

5.1.2 Subproblem for Total Variation along t

The second sub optimization problem of Eq (19) is the regularizer with a temporal Total Variation energy. In order to solve this problem, we need to minimize the following energy for the entire set of images $\mathbf{S} = \{S_0, S_1, ..., S_{t-1}\}$ collectively with the image consistency energy for the measurement $\mathbf{M} = \{M_0, M_1, ..., M_{t-1}\}$ and the sampling mask $\mathbf{R} = \{R_0, R_1, ..., R_{t-1}\}$ as shown below:

$$\min_{s} \theta \left\| \nabla_t \mathbf{S} \right\|_1 \quad s.t. \quad \left\| \mathbf{R}F\mathbf{S} - \mathbf{M} \right\|_2^2 < \varepsilon \tag{35}$$

By introducing the auxiliary variable $\mathbf{P}$, the above problem becomes:

$$\min_{\mathbf{P}} \theta \left\| \mathbf{P} \right\|_1 \quad s.t. \quad \left\| \mathbf{R}F\mathbf{S} - \mathbf{M} \right\|_2^2 < \varepsilon, \quad \nabla_t \mathbf{S} = \mathbf{P} \tag{36}$$

By adding dual variable $\mathbf{Q}$, it results in an unconstrained problem as following:

$$\min_{\mathbf{S},\mathbf{P}} \theta \left\| \mathbf{P} \right\|_1 + \frac{\gamma}{2} \left\| \mathbf{R}F\mathbf{S} - \mathbf{M} \right\|_2^2 + \frac{\delta}{2} \left\| \nabla_t \mathbf{S} - \mathbf{P} + \mathbf{Q} \right\|_2^2 \tag{37}$$

Solve for P:

$$\min_{\mathbf{P}} \theta \left\| \mathbf{P} \right\|_1 + \frac{\delta}{2} \left\| \nabla_t \mathbf{S} - \mathbf{P} + \mathbf{Q} \right\|_2^2 \tag{38}$$

$$\mathbf{P} = \mathcal{S}hrink_{\theta/\delta} \left(\nabla_t \mathbf{S} + \mathbf{Q} \right) \tag{39}$$

Solve for S:

$$\min_{\mathbf{S}} \frac{\gamma}{2} \left\| \mathbf{R}F\mathbf{S} - \mathbf{M} \right\|_2^2 + \frac{\delta}{2} \left\| \nabla_t \mathbf{S} - \mathbf{P} + \mathbf{Q} \right\|_2^2 \tag{40}$$

$$\left(\gamma F^H \mathbf{R}^H \mathbf{R} F - \delta \Delta_t \right) \mathbf{S} = \gamma F^H \mathbf{R}^H \mathbf{M} + \delta \nabla_t^H \left(\mathbf{P} - \mathbf{Q} \right) \tag{41}$$

Since the dimension of operator F is 2D but that of Δ_t is 1D, an efficient update rule for $\mathbf{S}$ can be obtained using a single iteration method as shown in [99].

Solve for Q: The update rule for $\mathbf{Q}$ is defined using a fixed point iteration as follows:

$$\mathbf{Q} = \mathbf{Q} + \left(\nabla_t \mathbf{S} - \mathbf{P} \right) \tag{42}$$

Note that the dual variable $\mathbf{Q}$ and others (U, H) are equivalent to Bregman variables used in [99], in term of terminology.

5.2 Result

In order to assess the performance of the proposed method, we ran our algorithm and [17] on four tumor DCE-MRI datasets. In the experiment, we used 2D atoms (i.e., filters) of size 16×16. The MRI datasets used in this experiment were perfusion dynamic tumor images processed with the Gd-DTPA agent to reconstruct the contrast profiles overtime. The CS-undersampling factor was set to ×8.

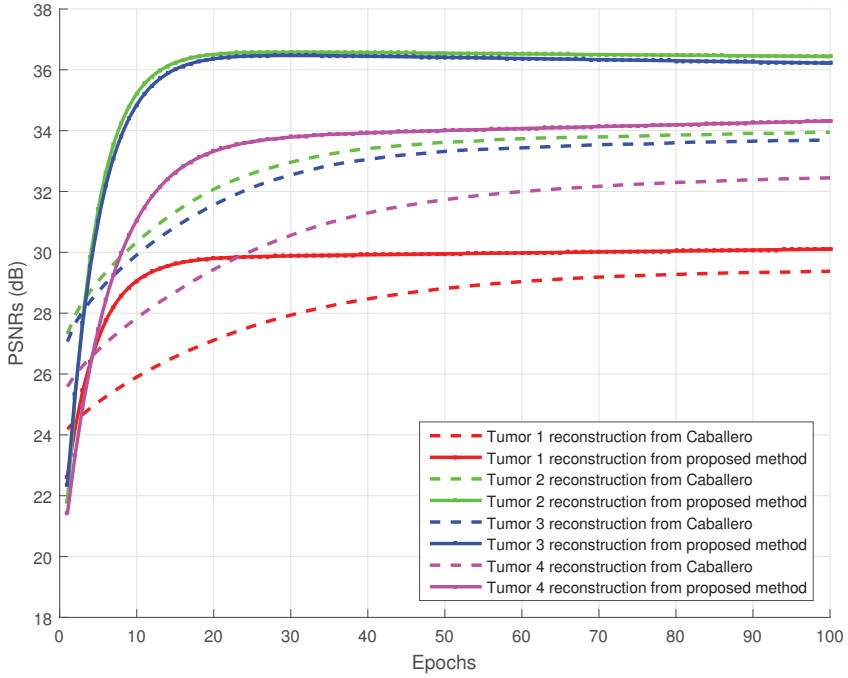


Figure 34: Convergence rate evaluation based on PSNRs.

Convergence evaluation: As showed in Figure 65, the achieved Peak Signal-To-Noise-Ratios (PSNRs) of the proposed method are significantly higher than those of Caballero et al. [17]. In addition, our approach also converged faster to the steady stage on both datasets although our initial filter values are chosen randomly while Caballero et al. used the discrete cosine transform (DCT) basis at the beginning. The difference results are mainly from the numerical solver and the energy formulation.

Running time evaluation: The wall-clock running times are measured on a PC equipped with an Intel i7 CPU with 16 GB main memory and an NVIDIA GTX Geforce 980 Ti GPU. The prototype code is written in MATLAB 2015b including GPU implementation. As shown in Figure 35, we observed that our method is about 7× to 9× faster than the stage-of-the-art dictionary learning based CS-MRI reconstruction method [17] for 100 epochs (i.e., the number of learning iterations).

Quality evaluation: Figure 38 visualizes the first 64 (out of 256) atoms, the reconstructed images, and the errors compared to the full reconstruction, respectively. Our learning method can generate atoms that capture details of the image features much better than the patch-based learning method (Fig. 38a). Because the patch-based learning method generate smoother atoms (e.g., Gabor-like edges), the reconstruction process relies more on accurate sparse coding, which results in less-optimal convergence in the minimization process and higher error rates compared to our method.

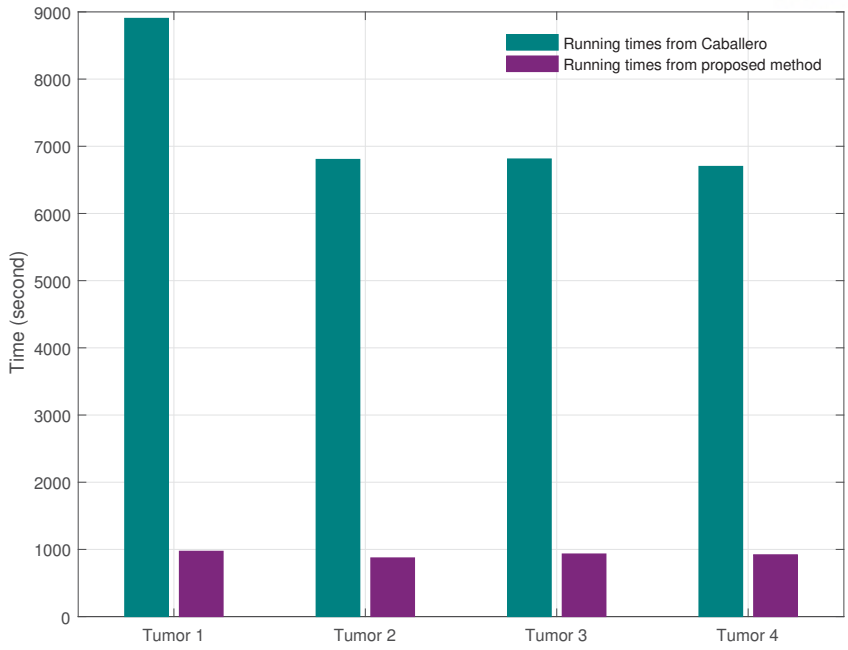
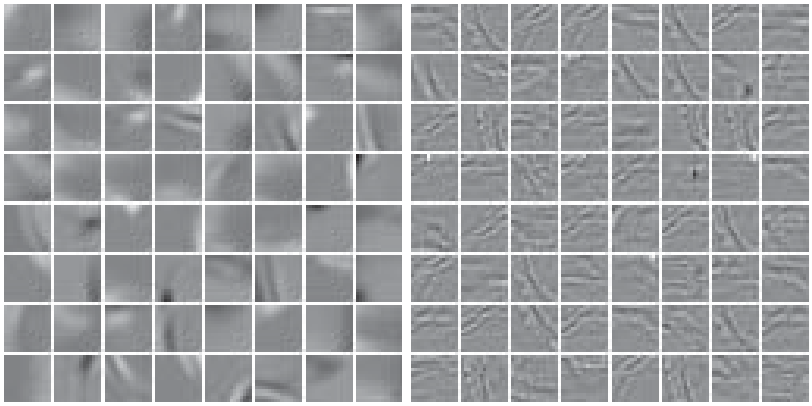


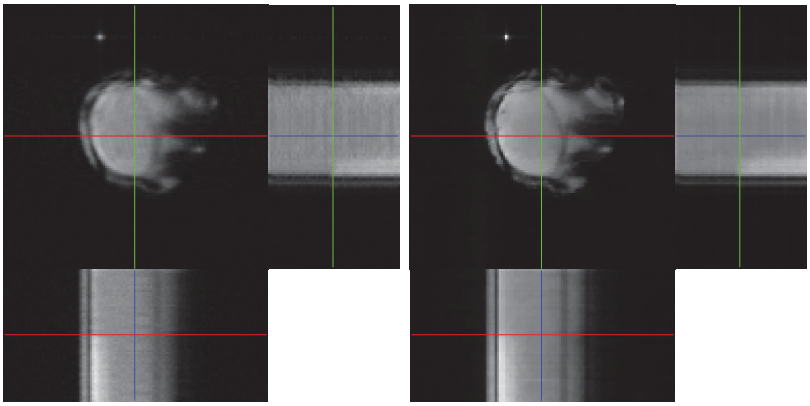
Figure 35: Running times of 100 epochs.

5.3 Summary

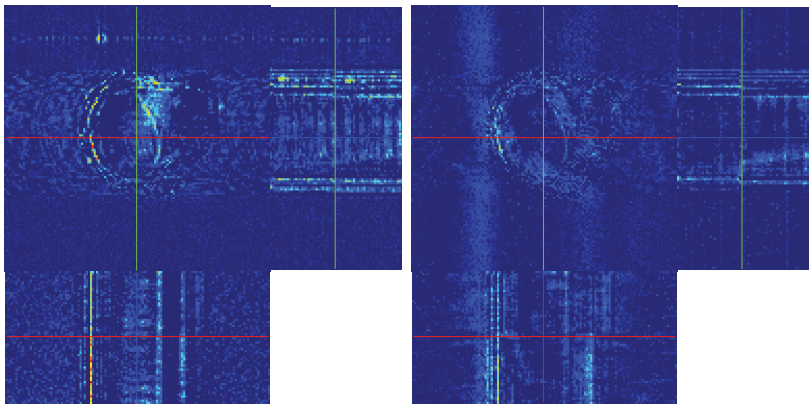
In this part, we introduced an efficient CS-MRI reconstruction method based on convolutional sparse coding and a temporal Total Variation regularizer. The proposed numerical solver is derived under the ADMM framework by leveraging the Fourier convolution theorem, which can be effectively accelerated using GPUs. As a result, we achieved faster convergence rate and higher PSNRs compared to the state-of-the-art CS-MRI reconstruction method using a patch-based dictionary learning. In the future, we plan to extend this method to include the temporal coherency directly on the atoms and assess its feasibility in clinical applications.



(a)



(b)



(c)

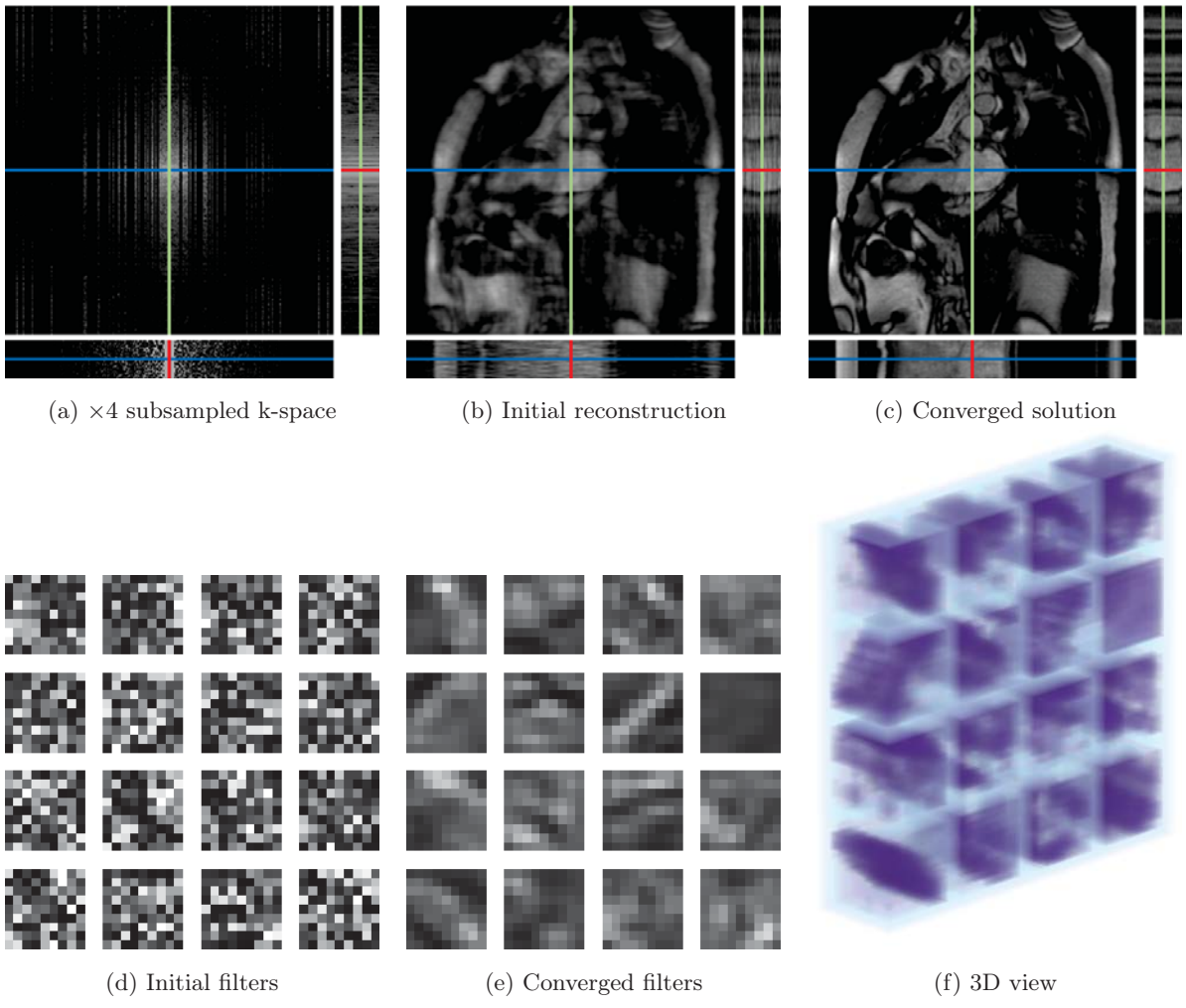
Figure 36: Left: the patch-based dictionary learning method [17]. Right: the proposed method. (a) generated dictionaries, (b) reconstructed images, (c) error plots (red: high, blue: low).

VI 3D Convolutional Sparse Coding Reconstruction of dCS-MRI

Dynamic cardiac MRI is considered as the gold standard among several imaging modalities in heart function diagnosis. However, due to its long acquisition time, its clinical application has been limited to non-time-critical ones. Recent research advances in Compressed Sensing (CS) [37] have been successfully applied to MRI [87] to reduce acquisition time. Nevertheless, CS-MRI poses a new challenge – the reconstruction time also increases because it needs to solve an in-painting inverse problem in the frequency domain (i.e., k-space). Therefore, accelerating the reconstruction process is a top priority to adopt CS framework to fast MRI diagnosis.

Conventional CS-MRI reconstruction methods have exploited the sparsity of signal by applying universal sparsifying transforms such as Fourier (e.g., discrete Fourier transform (DFT) or discrete cosine transform (DCT)), Total Variation (TV), and Wavelets (e.g., Haar, Daubechies, etc.). This research direction has focused on accelerating the sparsity-based energy minimization problem, with [99] or without hardware supports [70]. Some strategies were designed to accelerate the minimization process such as using TV plus nuclear norm [135] or proposed the solver in other sparsity domain such as low-rank technique [95, 123] More recently, the other approaches leveraging the state-of-the-art data-driven method, i.e., dictionary learning [1] (DL), have been proposed to further enhance the reconstruction quality [17, 19, 104**?**]. However, the existing learning-based methods suffer from the drawback of patch-based dictionary (i.e., redundant atoms and longer running times).

Convolutional sparse coding (CSC) is a new learning-based sparse representation that approximates the input signal with a superposition of sparse feature maps convolved with a collection of filters. This advanced technique replaces the patch-based dictionary learning process with an energy minimization process using a convolution operator on the image domain, which leads to an element-wise multiplication in frequency domain, derived within Alternating Direction Method of Multiplier (ADMM) framework [15], and later its direct inverse problem is introduced by Wohlberg [131]. CSC can generate much compact dictionaries due to its *shift-invatiant* nature of filters, and the pixel-wise computation in Fourier domain maps well to parallel architecture. However, such advanced machine learning approaches have not been fully exploited in CS-MRI literature yet. Therefore, in this work, we propose a novel CS dynamic MRI reconstruction that exploits the compactness and efficiency of 3D CSC. The proposed 3D CSC directly encodes both spatial and temporal features from dynamic cardiac 2D MRI using a compact set of 3D atoms (i.e., filters) without regularizers enforcing temporal coherence (e.g., total variation along the time axis). We also show that the proposed method maps well to



(a) ×4 subsampled k-space (b) Initial reconstruction (c) Converged solution

(d) Initial filters (e) Converged filters (f) 3D view

Figure 37: An overview of CS-MRI reconstruction using 3D CSC method.

data-parallel architecture, such as GPUs, for further accelerating its running time significantly, up to two orders of magnitude faster compared to the state-of-the-art CPU implementation of CS-MRI using patch-based dictionary learning. To the best of our knowledge, this is the first CS-MRI reconstruction method based on GPU-accelerated 3D CSC.

6.1 Method

Figure 37 is a pictorial description of the proposed method. If the inverse Fourier transform is directly applied to undersampled MRI k-space data (Figure 37a ×4 undersampling), the reconstructed images will suffer from artifacts (Figure 37b). The zero-filling reconstruction will serve as an initial guess for our iterative reconstruction process with randomly initialized filters, e.g., a collection of 16 atoms of size 9×9×9 as shown in Figure 37d. Then the image and filters are iteratively updated until they converge as shown in Figure 37c and e, f.

The proposed CS-MRI reconstruction algorithm is a process of finding s (i.e., a stack of 2D MR images for a given time duration) in the energy minimization problem defined as follows:

$$\begin{aligned} \min_{d,x,s} \ & \frac{\alpha}{2} \left\| s - \sum_k d_k * x_k \right\|_2^2 + \lambda \sum_k \|x_k\|_1 \\ & s.t. : \|RF_2 s - m\|_2^2 < \varepsilon^2, \ \|d_k\|_2^2 \leqslant 1 \end{aligned} \tag{43}$$

where d_k is the k-th filter (or atom in the dictionary) and x_k, is its corresponding sparse code for s. In Equation 43, the first term measures the difference between s and its sparse approximation $s - \sum_k d_k * x_k$, weighted by α. The second term is the sparsity regularization of x_k using an ℓ_1 norm with a weight λ instead of an ℓ_0 norm as used in [1, 17, 20]. The rest of the equation is the collection of constraints - the first constraint enforces the consistency between undersampled measurement m and the undersampled reconstructed image using the mask R and the Fourier operator F, and the second constraint restricts the Frobenius norm of each atom d_k within a unit length. In the following discussion, we will use a simplified notation without indices k and replace the result of Fourier transform of a given variable by using the subscript f (for example, d_f is the simplified notation for Fd in 3D domain and s_{f_2} is the simplified notation for $F_2 s$ in 2D spatial domain) to derive the solution of Equation 43. Therefore, problem 43 can be rewritten using auxiliary variables y and g for x and d as follows:

$$\begin{aligned} \min_{d,x,g,y,s} \ & \frac{\alpha}{2} \left\| s - \sum d * x \right\|_2^2 + \lambda \|y\|_1 \\ & s.t. : x - y = 0, \ \|RF_2 s - m\|_2^2 < \varepsilon^2, g = \mathcal{P}roj(d), \ \|g\|_2^2 \leqslant 1 \end{aligned} \tag{44}$$

where g and d are related by a projection operator as a combination of a truncated matrix followed by a padding-zero matrix in order to make the dimension of g same as that of x. Since we will leverage Fourier transform to solve this problem, g should be zero-padded to make its size same as g_f and x_f. The above constrained problem can be rebuilt in an unconstrained form with dual variables u, h, and further regulates the measurement consistency and the dual differences with γ, ρ, and σ, respectively:

$$\begin{aligned} \min_{d,x,g,y,s} \ & \frac{\alpha}{2} \left\| s - \sum d * x \right\|_2^2 + \lambda \|y\|_1 + \frac{\gamma}{2} \|RF_2 s - m\|_2^2 + \frac{\rho}{2} \|x - y + u\|_2^2 + \frac{\sigma}{2} \|d - g + h\|_2^2 \\ & s.t. : g = \mathcal{P}roj(d), \ \|g\|_2^2 \leqslant 1 \end{aligned} \tag{45}$$

Then we can solve problem 45 by iteratively finding the solution of independent smaller problems, as described below:

Solve for x:

$$\min_{x} \ \frac{\alpha}{2} \left\| \sum d * x - s \right\|_2^2 + \frac{\rho}{2} \left\| x - y + u \right\|_2^2 \tag{46}$$

If we apply the Fourier transform to the 46, it becomes:

$$\min_{x_f} \ \frac{\alpha}{2} \left\| \sum d_f x_f - s_f \right\|_2^2 + \frac{\rho}{2} \left\| x_f - y_f + u_f \right\|_2^2 \tag{47}$$

Then the minimum solution of 47 can be found by taking the derivative of 47 with respect to x_f and setting it to zero as follows:

$$\left(\alpha D_f^H D_f + \rho I\right) x_f = D_f^H s_f + \rho \left(y_f - u_f\right) \tag{48}$$

Note that the notation D_f stands for the concatenated matrix of all diagonalized matrices d_{fk} as follows: $D_f = [diag(d_{f1}), ..., diag(d_{fk})]$ and D_f^H is the complex conjugated transpose of D_f.

Solve for y:

$$\min_{y} \quad \lambda \|y\|_1 + \frac{\rho}{2} \left\| x - y + u \right\|_2^2 \tag{49}$$

y for ℓ_1 minimization problem can be found by using a shrinkage operation:

$$y = \mathcal{S}_{\lambda/\rho} \left(x + u\right) \tag{50}$$

Update for u: The update rule for u can be defined as a fixed- point iteration with the difference between x and y (u converges when x and y converge each other) as follows:

$$u = u + x - y \tag{51}$$

Solve for d:

$$\min_{d} \quad \frac{\alpha}{2} \left\| \sum d * x - s \right\|_2^2 + \frac{\sigma}{2} \left\| d - g + h \right\|_2^2 \tag{52}$$

Similar to x, d can be solved in the Fourier domain:

$$\min_{d_f} \quad \frac{\alpha}{2} \left\| \sum d_f x_f - s_f \right\|_2^2 + \frac{\sigma}{2} \left\| d_f - g_f + h_f \right\|_2^2 \tag{53}$$

$$\left(\alpha X_f^H X_f + \sigma I\right) d_f = X_f^H s_f + \sigma \left(g_f - g_f\right) \tag{54}$$

where X_f stands for the concatenated matrix of all diagonalized matrices x_{fk} as follows: $X_f = [diag(x_{f1}), ..., diag(x_{fk})]$ and X_f^H is the complex conjugated transpose of X_f.

Solve for g:

$$\min_{g} \quad \frac{\sigma}{2} \left\| d - g + h \right\|_2^2 \qquad s.t.: \qquad g = \mathcal{P}roj(d), \ \|g\|_2^2 \leqslant 1 \tag{55}$$

g can be found by taking the inverse Fourier transform of d_f. This projection should be constrained by suppressing the elements which are outside the filter size d_k, and followed by normalizing its ℓ_2-norm to a unit length.

Update for h: Similar to u, the update rule for h can be defined as follows:

$$h = h + d - g \tag{56}$$

Solve for s:

$$\min_{s} \quad \frac{\alpha}{2} \left\| s - \sum d * x \right\|_2^2 + \frac{\gamma}{2} \left\| RF_2 s - m \right\|_2^2 \tag{57}$$

The objective function of 57 can be transformed into 2D Fourier domain:

$$\min_{s_{f_2}} \quad \frac{\alpha}{2} \left\| s_{f_2} - F_t^H \sum d_f x_f \right\|_2^2 + \frac{\gamma}{2} \left\| R s_{f_2} - m \right\|_2^2 \tag{58}$$

Since d_f and x_f obtained previously in 3D Fourier domain, we need to bring it onto the same space by applying an inverse Fourier transform along time-axis F_t^H. Then s_{f_2} can be found by solving the following linear system:

$$\left(\gamma R^H R + \alpha I\right) s_{f_2} = \gamma R^H m + \alpha F_t^H \sum d_f x_f \tag{59}$$

Note that the efficient solutions of 48, 54 and 59 can be determined via the Sherman-Morrison formula for independent linear systems as shown in [131]. To this end, after the iteration process, s will be the results of applying a 2D inverse Fourier transform F_2^H on s_{f_2}.

Implementation details: Since the above derivation consists only Fourier transform and element-wise operations, it maps well to data-parallel architecture, such as GPUs. We used MATLAB to implement the proposed method using the GPU. We set $\alpha = 1$, $\gamma = 1$, $\lambda = 0.1$, $\rho = 10$, $\sigma = 10$ and keep refining the filter banks as well as the reconstruction iteratively until they converge.

6.2 Result

In order to assess the performance of the proposed method, we compared our algorithm with the stage-of-the-art dictionary learning-based CS reconstruction from Caballero et. al. [17], and the conventional CS reconstruction using wavelet and total variation energy from Quan et. al. [99]. We used three cardiac MRI datasets from The Data Science Bowl [11] – 2 chamber view (2ch), 4 chamber view (4ch), and short axis view (sax). Each dataset consists of 30 frames of a 256×256

[11] `https://www.kaggle.com/c/second-annual-data-science-bowl/data`

image across the cardiac cycle of a heart. In the experiment, we used 3D atoms of size 9×9×9 and CS-undersampling factor was set to ×4.

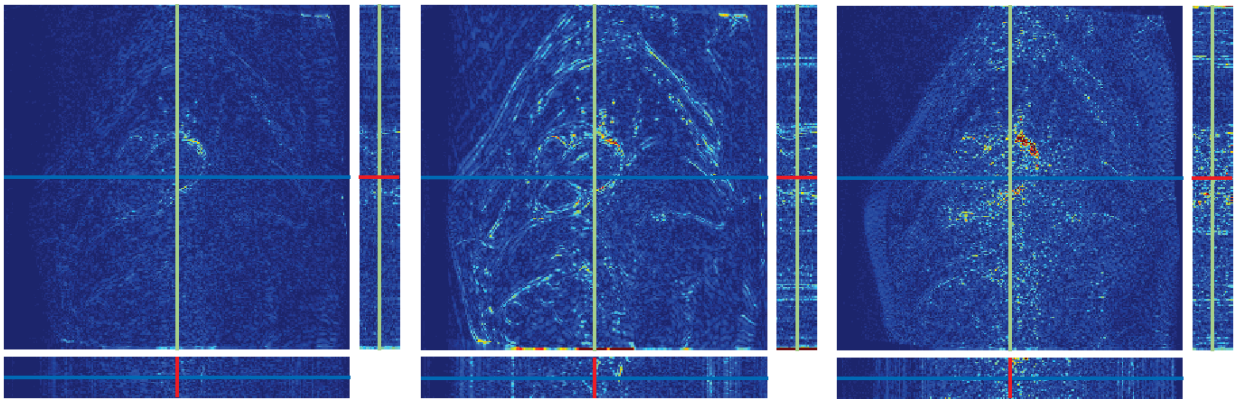
Running time evaluation: In order to make this direct performance comparison of learning-based methods between the proposed one and Caballero et al. [17], we measured wall clock running time of both methods on a PC equipped with an Intel i7 CPU with 16 GB main memory and an NVIDIA GTX Geforce Titan X GPU. Our prototype code is written in MATLAB 2015b including GPU implementation, and we used the author-provided MATLAB code for Caballero et al. [17]. As shown in Table 12, we observed that our CPU-based method is about 54× to 73×, or about two orders of magnitude, faster than the stage-of-the-art DL-based CS-MRI reconstruction method for 100 epochs (i.e., the number of learning iterations). In addition, our GPU-based accelerated implementation also outperforms the CPU version about 1.25× to 3.82×, which is greatly reduced to a level closer to be ready for clinical application. We expect that the performance of our method can improve further by using CUDA C/C++ without MATLAB.

Table 12: Reconstruction times of learning-based methods (100 epochs)

Methods	2ch	4ch	sax
Caballero et al. [17]	558 Min	475 Min	427 Min
Our method (CPU)	7.67 Min	7.73 Min	7.94 Min
Our method (GPU)	6.14 Min	2.70 Min	2.08 Min

Quality evaluation: Figure 38 visualizes the reconstruction errors compared to the full reconstruction of each method, respectively. As can be seen, our approach generated less error compared to the stage-of-the-art method of [17] and conventional CS-reconstruction using wavelet and TV energy [99]. Their glitches on the temporal profile are clearly observed since total variation along time axis may smooth out the temporal features that move quickly, especially near the heart boundary. In our case, the learned atoms are in 3D with larger supports, which helps to capture the time trait better even under fast motion and reduces errors in the reconstructed images. In addition, shift-invariance of CSC helps to generate more compact filters compared to the patch-based method.

Figure 65 shows the achieved PSNRs measured between the CS-reconstruction results and the full reconstruction. As shown in this figure, our method requires more iterations (epochs) to converge to the steady state, but the actual running time is much faster than the others due to GPU acceleration. In the mean time, our method can reach much higher PSNRs.



(a) (b) (c)

Figure 38: Error plots (red: high, blue: low) between full reconstruction and the result from (a) the proposed method, (b) the DL-based method [17], and (c) wavelet and total variation energy method [99].

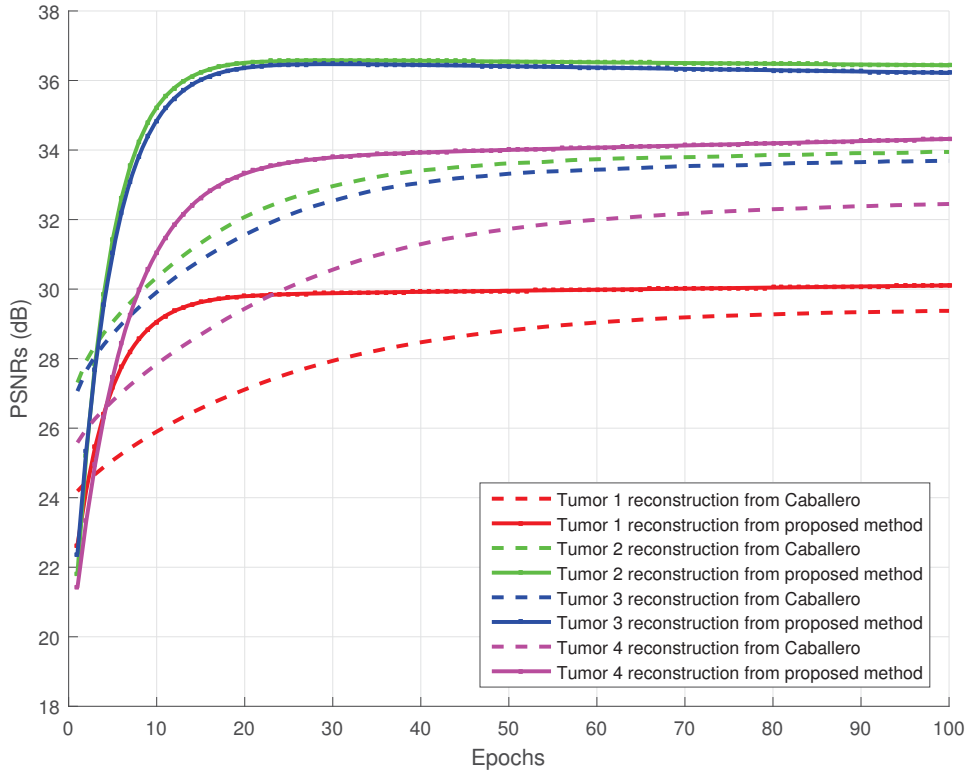


Figure 39: Convergence rate evaluation based on PSNRs.

6.3 Summary

In this work, we introduced an efficient CS-MRI reconstruction method based on pure 3D convolutional sparse coding where shift-invariant 3D filters can represent the temporal features of the MRI data. The proposed numerical solver is derived under the ADMM framework by leveraging the Fourier convolution theorem, which can be effectively accelerated using GPUs. As a result, we achieved faster running time and higher PSNRs compared to the state-of-the-art CS-MRI reconstruction methods, such as using a patch-based dictionary learning and conventional wavelet and total variation energy. In the future, we plan to conduct a proper controlled-study of tuning-parameters and assess its feasibility in clinical applications.

VII GAN-based Reconstruction of CS-MRI with Cyclic Loss

7.1 Method

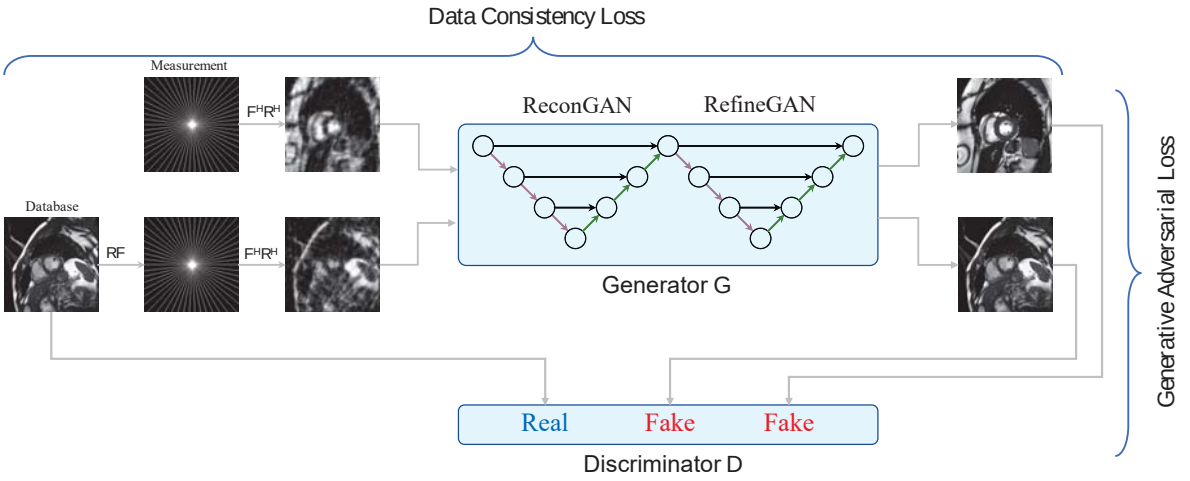


Figure 40: Overview of the proposed method: it aims to reconstruct the images which are satisfied the constraint of under-sampled measurement data; and whether those look similar to the fully aliasing-free results. Additionally, if the fully sampled images taken from the database go through the same process of under-sampling acceleration; we can still receive the reconstruction as expected to the original images.

7.1.1 Problem Definition and Notations

We denote the under-sampled raw MRI data (k-space measurements) using the sampling mask R as m. Then, its zero-filling reconstruction s_0 can be obtained by the following equation:

$$s_0 = F^H R^H (m) \tag{60}$$

where F is the Fourier operator, and superscript H indicates the conjugated transpose of a given operator. Similarly, turning any image s_i into its under-sampled measurement m_{s_i} with the given sampling mask R can be done via the inverse of the reconstruction process:

$$m_{s_i} = RF (s_i) \tag{61}$$

Then, compressed sensing MRI reconstruction, which is a process of generating a full-reconstruction image s from under-sampled k-space data m, can be described as follows:

$$\min_s J (s) \quad s.t. \quad RF(s) = m \tag{62}$$

where $J (s)$ is a regularizer required for ill-posed optimization problems. In our method, this energy minimization process is replaced by the training process of the neural network.

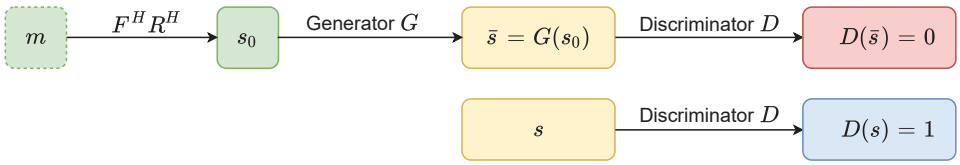


Figure 41: Two learning processes are trained adversarially to achieve better reconstruction from generator G and to fool the ability of recognizing the real or fake MR image from discriminator D.

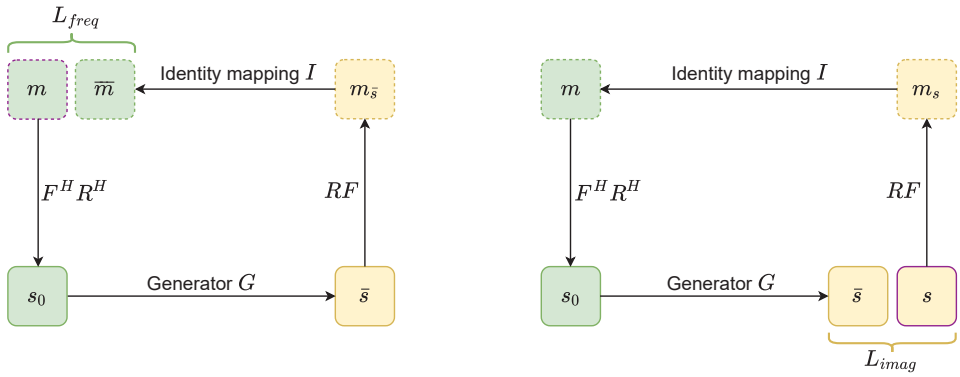


Figure 42: The cyclic data consistency loss, which is a combination of under-sampled frequency loss and the fully reconstructed image loss.

7.1.2 Overview of the Proposed Method

Figure 40 represents an overview of the proposed method: Our generator G consists of two-fold chained networks that generate the full MR image directly from a zero-filling reconstruction image (i.e., image generated from under-sampled k-space data), in which each input can be up to 2-channel to represent real- and imaginary image of the complex-valued MRI data. The generated result is favorable to the fully-sampled data taken from an extensive database and put through the same under-sampling process. In contrast, the discriminator D attempts to differentiate between the real MRI instances from the database and the fake results output generated by G. The entire system involves training G and D adversarially until a balance is reached at the convergence stage. Details of each component will be discussed shortly.

7.1.3 Generative Adversarial Loss

Our objective is to train generator G, which can transform any zero-filling reconstruction $s_0 = F^H R^H(m)$, $m \in M$, where M is the collection of under-sampled k-space data, to a fully-reconstructed image $\overline{s}$ under the constraint that $\overline{s}$ is *indistinguishable* from all images $s \in S$ reconstructed from full k-space data. To accomplish this aim, a discriminator D is attached to

distinguish whether the image is synthetically generated from s_0 by G ($\overline{s}$, which is considered *fake*) or is reconstructed from fully-sampled k-space data (s, which is considered *real*). In other words, at each epoch, G tries to produce a reconstruction that can fool D whereas D avoids to be fooled. This kind of training borrows the win-lose strategy which is very common in game theory. We wish to train D so that it can maximize the probability of assigning the correct *true* or *false* label to images. Note that the objective function for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability, where the image comes from: $D(\overline{s}) = D(G(s_0)) = 0$ (*fake*), and $D(s) = 1$ (*real*). Simultaneously, generator G is trained to minimize $[1 - \log D(\overline{s})]$ or $[1 - \log D(G(s_0))]$. This can be addressed by formally defining an adversarial loss L_{adv}, for which we wish to find the solutions of its minimax problem:

$$\min_G \max_D L_{adv}(G, D) \tag{63}$$

where

$$L_{adv}(G, D) = \underset{m \in M}{E}[1 - \log D(G(s_0))] + \underset{s \in S}{E}[\log D(s)] \tag{64}$$

Figure 41 is a schematic depiction of our adversarial process: G tries to generate images $\overline{s} = G(s_0)$ look similar to the images s that have been reconstructed from full k-space data, while D aims to distinguish between $\overline{s}$ and s. Once the training converges, G can produce the result $\overline{s}$ that is close to s, and D is unable to differentiate between them, which results in bringing the probability for both real and fake labels to 50%. In practice, Wasserstein GAN [4] energy is commonly used to improved the stability of the training processes and is also adopted to our method.

7.1.4 Cyclic Data Consistency Loss

In an extreme case, with large enough resources and data, the network can map the zero-filling reconstruction s_0 to any existing fully reconstructed images $s \in S$. Therefore, the adversarial loss alone is not sufficient to correctly map the under-sampled data $s_0[n]$ and the full reconstruction $\overline{s}[n]$ for all n. To strengthen the bridge connection between $s_0[n]$ and $\overline{s}[n]$, we introduce an additional constraint, the data consistency loss L_{cyc}, which is a combination of under-sampled frequency loss L_{freq} and fully reconstructed image loss L_{imag} in a cyclic fashion. The first term L_{freq} guarantees that when we perform another under-sampled operator RF on reconstructed images $\overline{s}[i]$ to get $\overline{m}[i]$, the difference between $\overline{m}[i]$ and $m[i]$ should be minimal. The second energy term L_{imag} ensures that for any other images $s[j] \in S$ taken from the fully reconstructed data, if $s[j]$ goes through the under-sampling process (by applying RF), and the generator G takes zero-filling reconstruction $s_0[j]$ (by applying $F^H R^H$ to $m[j]$) to produce the reconstruction

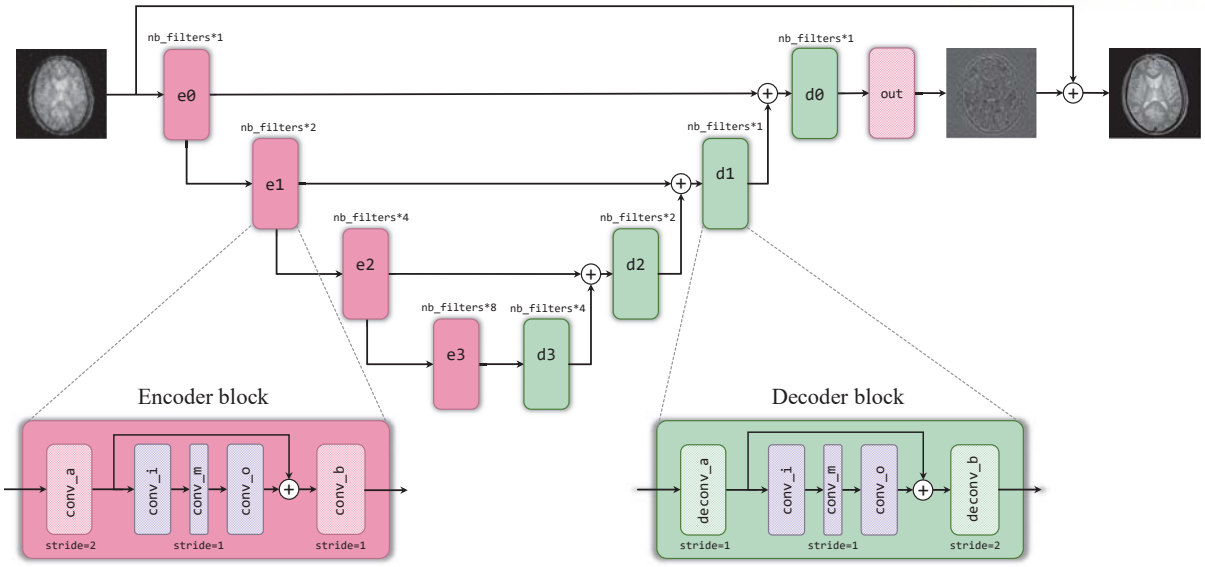
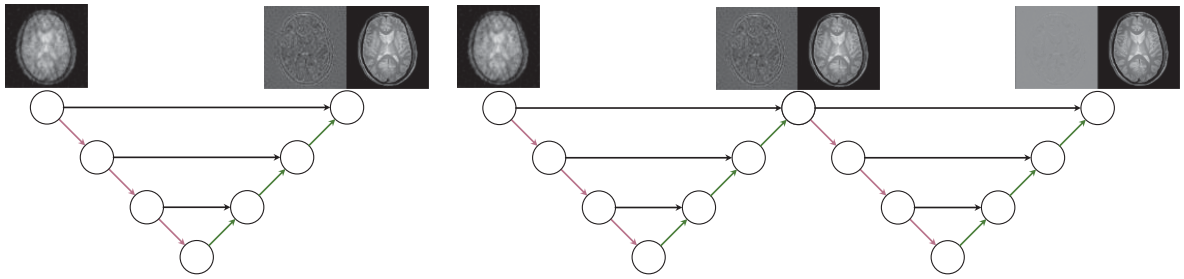


Figure 43: Generator G, built by basic building blocks, can reconstruct inverse amplitude of the residual component causes by reconstruction from under-sampled k-space data. The final result is obtained by adding the zero-filling reconstruction to the output of G



(a) ReconGAN (b) RefineGAN

Figure 44: One-fold (a) and two-fold architectures (b) of the generator G.

$\overline{s}[j]$, then both $\overline{s}[j]$ and $s[j]$ should appear to be similar. Those two losses are described in a cyclic fashion in Figure 42. In practice, various distance metrics, such as mean-square-error (MSE), mean-absolute-error (MAE), etc, can be employed to implement L_{cyc}:

$$L_{cyc}(G) = L_{freq}(G) + L_{imag}(G) = \mathbf{d}(m[i], \overline{m}[i]) + \mathbf{d}(s[j], \overline{s}[j]) \tag{65}$$

Note that the data consistency loss only affects the generator G and not the discriminator D. Furthermore, each individual loss L_{freq} or L_{imag} is evaluated on its own samples: $m[i]$ and $s[j]$ are drawn independently from M and S. Since they are non-complex pixel-wise distance metrics, they will return scalars regardless the input data are either magnitude- or complex-valued numbers.

7.1.5 Model Architecture

In this section, we introduce the details of our neural network architecture, which is a variant of a convolutional autoencoder and deep residual network.

Fundamental blocks To begin discussing the model architecture, we first introduce three fundamental components in our generative adversarial model: *encoder*, *decoder*, with the insertion of the *residual* block. The details of each block are described pictorially in Figure 43. The *encoder* block, shaded in red, accepts a 4D tensor input and performs 2D convolution with `filter_size` 3×3, and `stride` is equal to 2 so that it performs down-sampling with convolution without a separate max-pooling layer. The number of feature maps `filter_number` is denoted in the top. The *decoder* block, shaded in green, functions as the convolution transpose, which enlarges the resolution of the 4D input tensor by two times. The *residual* block, shaded in violet, is used to increase the depth of generator G and discriminator D networks, and it consists of three convolution layers: the first layer `conv_i` with `filter_size` is 3×3, `stride` is 1 and the number of feature maps is `nb_filters`, which reduces the dimension of the tensor by half. The second layer `conv_m` performs the filtering via a 3×3 convolution with the same number of feature maps `nb_filters/2`. The remaining `conv_o`, which has `filter_size` 3×3, `stride` is 1 and `nb_filters`, feature maps, retrieves the input tensor shape so that they can be combined to form a residual bottleneck. This *residual* block allows us to effectively construct a deeper generator G and discriminator D without suffering from the gradient vanishing problem [57].

Generator architecture Figure 43 illustrates the architecture of our generator G. It is built based upon the design of a convolutional autoencoder, which consists of an encoding path (left half of the network) to retrieve the compressed information in latent space, and a symmetric decoding path (right half of the network) that enables the prediction of synthesis. The convolution mode we used is "same", which leads the final reconstruction to have a size identical to the input images. The encoding and decoding paths consist of multiple levels, i.e., image resolutions, to extract features across different scales. Three types of introduced building blocks (i.e., *encoder*, *residual*, *decoder* and their related parameters) are used to construct the proposed generator G.

It is worth noting that the proposed generator G does not attempt to reconstruct the image directly. Instead, it is trained to produce the inverse amplitude of the noise caused by reconstruction from under-sampled data. The final reconstruction is obtained by adding the zero-filling input to the output from the generator G, which is similar to other current machine learning-based CS-MRI methods [79, 91, 134].

Discriminator architecture For the discriminator D, we use an architecture identical to that of the encoding path of the generator G. The output of the last *residual* block is used to evaluate the mentioned adversarial loss L_{adv} (Equation 64). To reiterate, if the discriminator receives the image $s \in S$, it will result in $D(s) = 1$, as a true result. Otherwise, the reconstruction $\overline{s}$ will be recognized as a fake result, or equivalently, $D(\overline{s}) = D(G(s_0)) = 0$.

Full Objective Function and Training Specifications In summary, our system involves two sub-networks which are trained adversarially to minimize the following loss:

$$L_{total} = L_{adv}(G, D) + \alpha L_{freq}(G) + \gamma L_{imag}(G) \tag{66}$$

where α and γ are the weights which help to control the balance between each contribution. We set $\alpha = 1.0$ and $\gamma = 10.0$ for all the experiments. The Adam optimizer [76] is used with the initial learning rate of $1e^{-4}$, which decreases monotonically over 500 epochs. The entire framework was implemented using a system-oriented programming wrapper (tensorpack [12]) of the tensorflow [13] library.

Chaining with Refinement Network The proposed generator G by itself can perform an end-to-end reconstruction from the zero-filling MRI to the final prediction. However, in the real-world setup, many iterative methods also take extra steps to go through the current result and then attempt to "correct" the mistakes. Therefore, we introduce an additional step in refining the reconstruction by concatenating a chain of multiple generators to resolve the ambiguities of the initial prediction from the generator. For example, Figure 44 shows the two-fold chaining generator in a self-correcting form. By forcing the desired ground-truth between them, the entire solution becomes a target-driven approach. This enables our method to be a single-input and multi-output model, where each checkpoint in between attempts to produce better a reconstruction. Because the architecture of each sub-generator is the same, we can think of the proposed model is another variant of the recurrent neural network that treats the entire sub-generator as a single state without sharing the weights after unfolding. The loss training curves of the checkpoints decrease as the number of checkpoints increases, and they eventually converge as the number of training iterations (epochs) increases. Interestingly, our generator can be considered a V-cycle in the multigrid method that is commonly used in numerical analysis, where the contraction in the encoding path is similar to *restriction* heads from fine to coarse grid. The expansion in the decoding path spans along the *prolongation* toward the final reconstruction, and the skip connections act as the *relaxation*. We refer to the first check point as *ReconGAN*

[12] `http://tensorpack.readthedocs.io/`

[13] `http://www.tensorflow.org/`

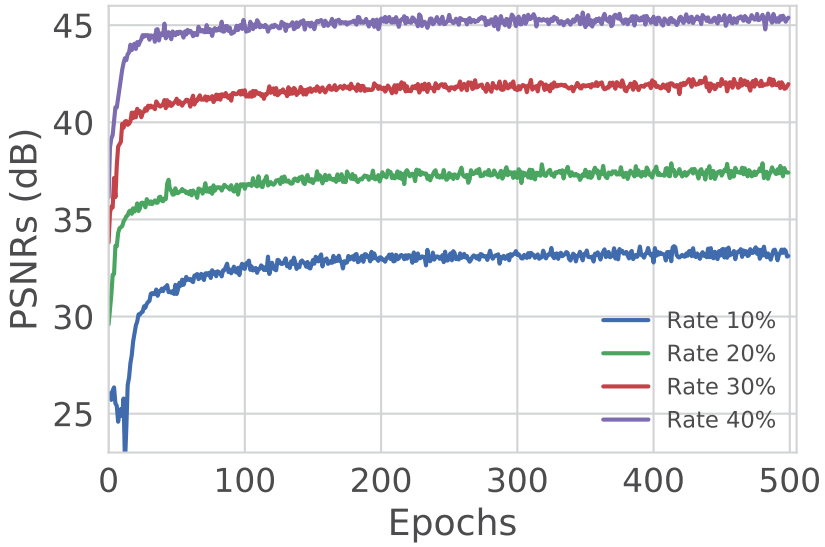


Figure 45: PSNR curves of RefineGAN with different undersampling rates on the brain training set over 500 epochs.

and the second check point in our two-fold chaining network as *RefineGAN*. The entire chaining structure (with 2 generators) is trained together. Each individual cycle has its own variable scope and hence, their weight are updated differently. The later structure serves as a boosting module which improves results.

7.2 Result

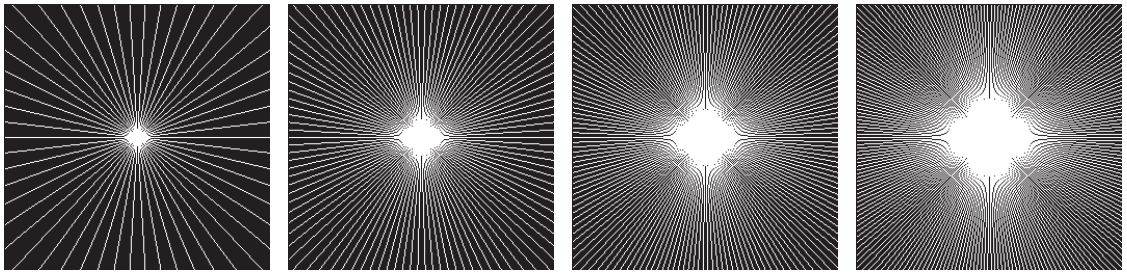
7.2.1 Results on real-valued MRI data

We trained many versions of the proposed networks with different factors of under-sampling masks (10%, 20%, 30%, 40%) on the training sets. As shown in Figure 45, those trained models (for RefineGAN) reach a convergence stage in a few hundred training iterations. The performances on the test sets are expected to show similar results.

We used two sets of MR images from the IXI database[14] (the brain dataset) and from the Data Science Bowl challenge [15] (the chest dataset) to assess the performance of our method by comparing our results with state-of-the-art CS-MRI methods (e.g., Convolutional sparse coding-based [101, 102], patch-based dictionary [18, 19, 104], deep learning-based [79, 110, 117, 128], and GAN-based [91, 134]). The image resolution of each image is 256x256. From each database,

[14] http://brain-development.org/ixi-dataset/

[15] https://www.kaggle.com/c/second-annual-data-science-bowl/data



(a) 10% (b) 20% (c) 30% (d) 40%

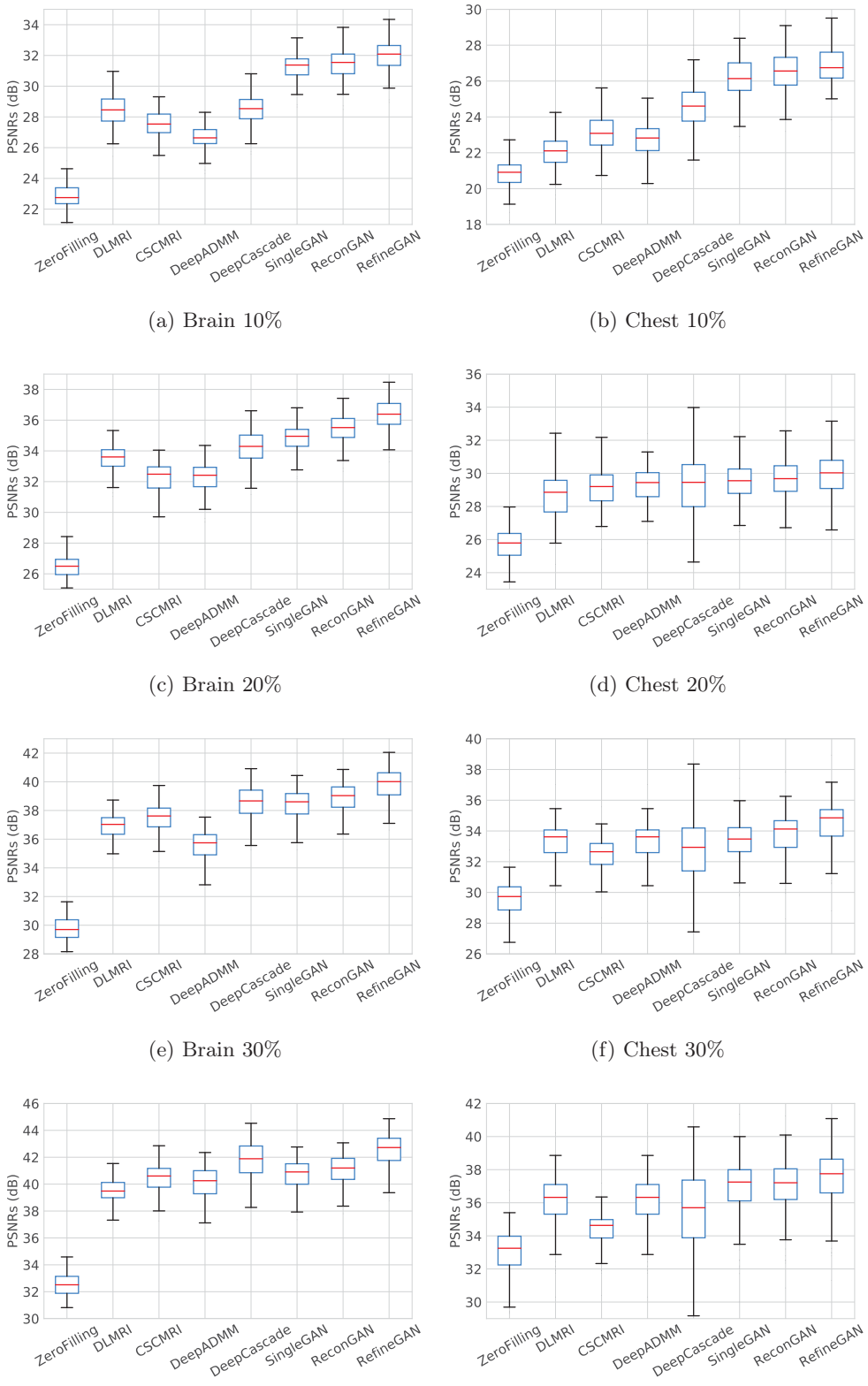
Figure 46: Radial sampling masks used in our experiments.

Table 13: Running time comparison of various CS-MRI methods (in seconds).

Abbreviation	Methods	Brain	Chest
CSCMRI	[101, 102]	8.56808	9.37082
DLMRI	[18, 19, 104]	604.24623	613.84531
DeepADMM	[117]	0.31725	0.28677
DeepCascade	[110]	0.22182	0.25627
SingleGAN	[91, 134]	0.064599	0.075529
ReconGAN	—	0.060753	0.068871
RefineGAN	—	0.106157	0.111607

we randomly selected 100 images for training the network and another 100 images for testing (validating) the result. We conducted the experiments for various sampling rates (i.e., 10%, 20%, 30%, and 40% of the original k-space data), corresponding to 10×, 5×, 3.3×, and 2.5× factors of acceleration. We assume the target MRI data type is static, and radial sampling masks are applied (Figure 46). It is worth noting that our experimental data are real-valued MRI images, which require pre-processing of the actual acquisition from the MRI scanner because the actual MRI data is complex-valued. Additional data preparation steps, such as data range normalization and imaginary channel concatenation, are also required.

Running Time Evaluation: Table 13 summarizes the running times of our method and other state-of-the-art learning-based CS-MRI methods. Even though dictionary learning-based approaches leverage pre-trained dictionaries, their reconstruction time depends on the numerical methods used. For example, CSCMRI by Quan and Jeong [101, 102] employed a GPU-based ADMM method, which is considered one of the state-of-the-art numerical methods, but the running time is still far from interactive (about 9 seconds). Another type of dictionary learning-



(a) Brain 10%

(b) Chest 10%

(c) Brain 20%

(d) Chest 20%

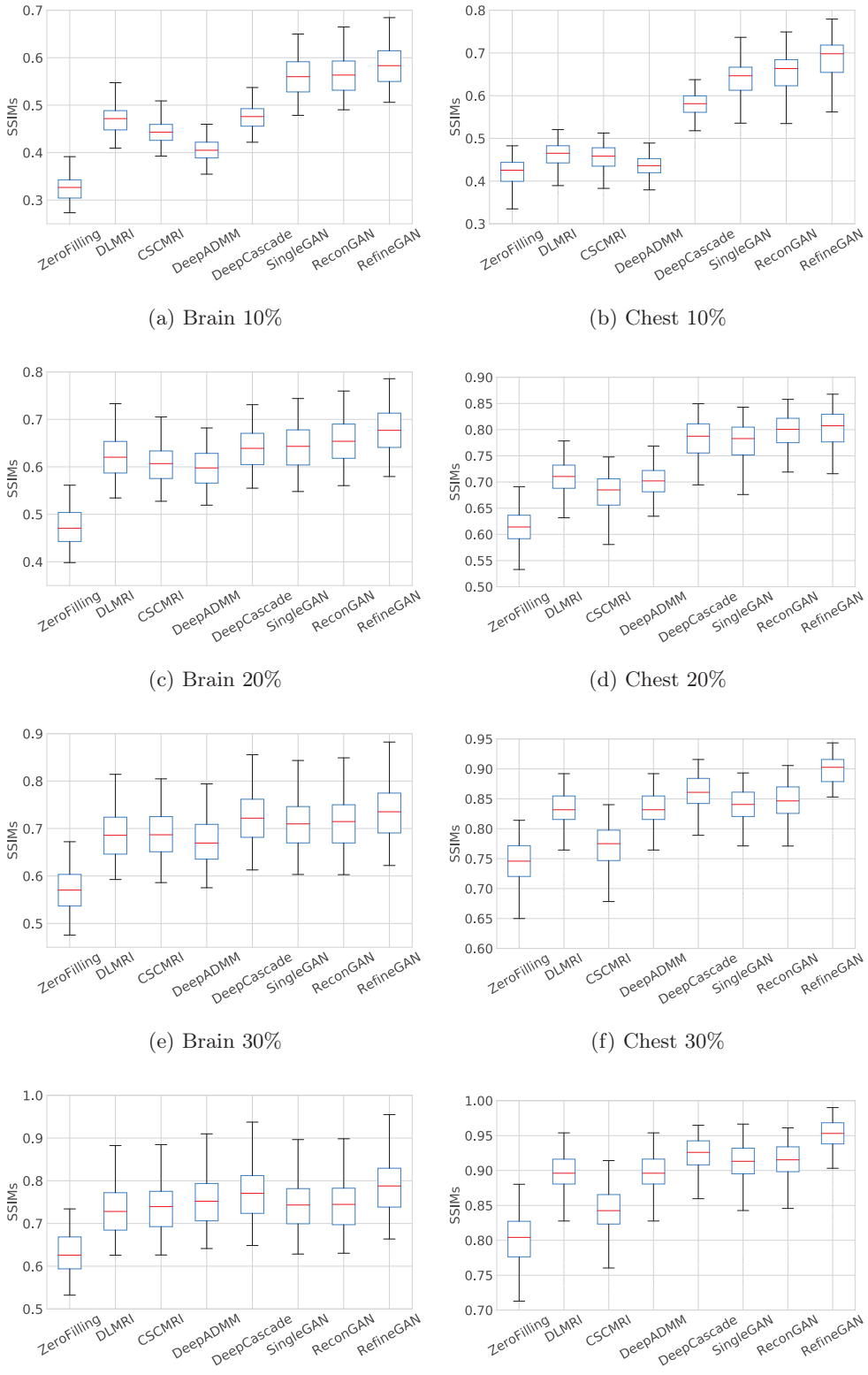
(e) Brain 30%

(f) Chest 30%

(g) Brain 40%

(h) Chest 40%

Figure 47: PSNRs evaluation on the brain and chest test set. Unit: dB



(a) Brain 10%

(b) Chest 10%

(c) Brain 20%

(d) Chest 20%

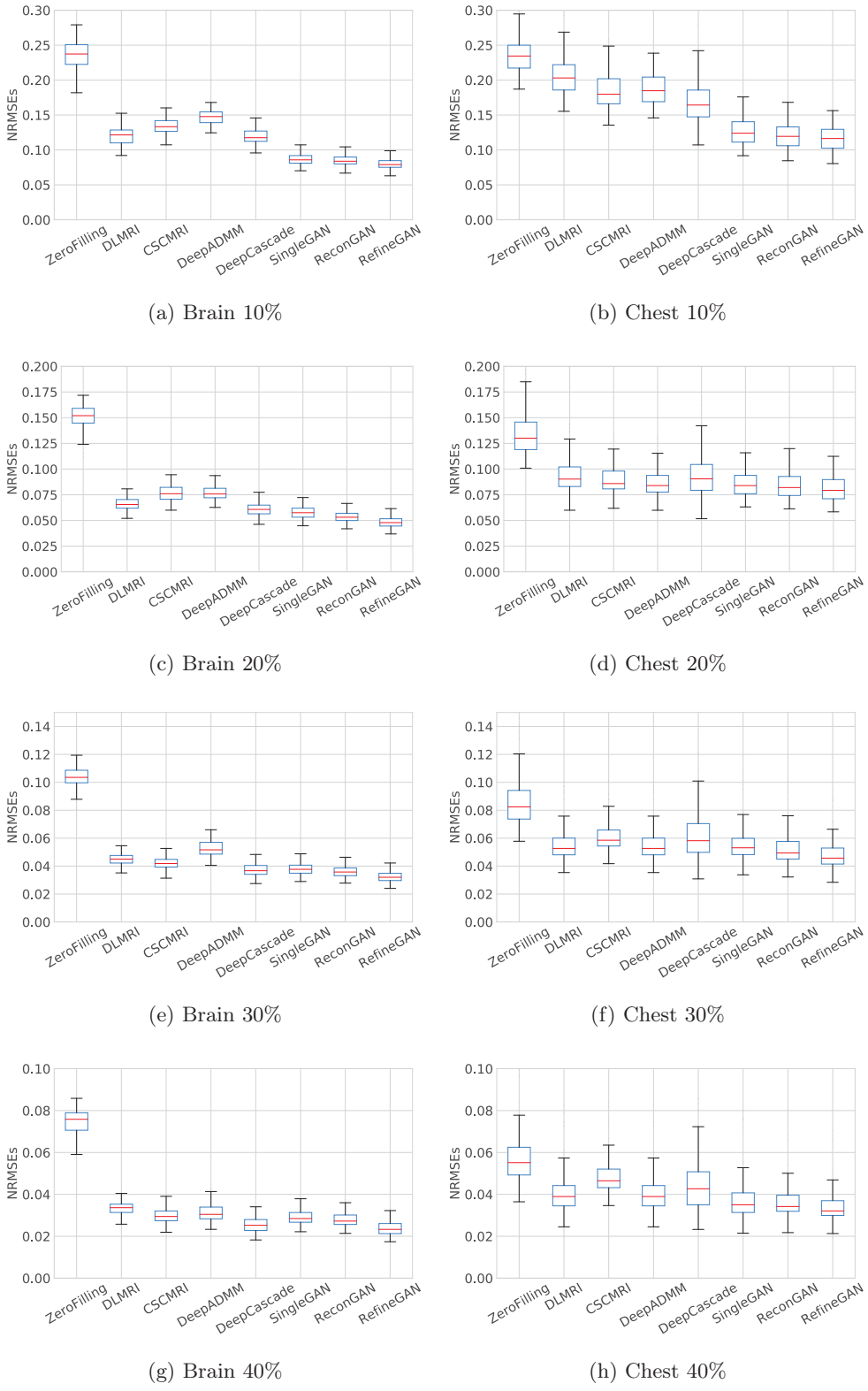
(e) Brain 30%

(f) Chest 30%

(g) Brain 40%

(h) Chest 40%

Figure 48: SSIMs evaluation on the brain and chest test set



(a) Brain 10%

(b) Chest 10%

(c) Brain 20%

(d) Chest 20%

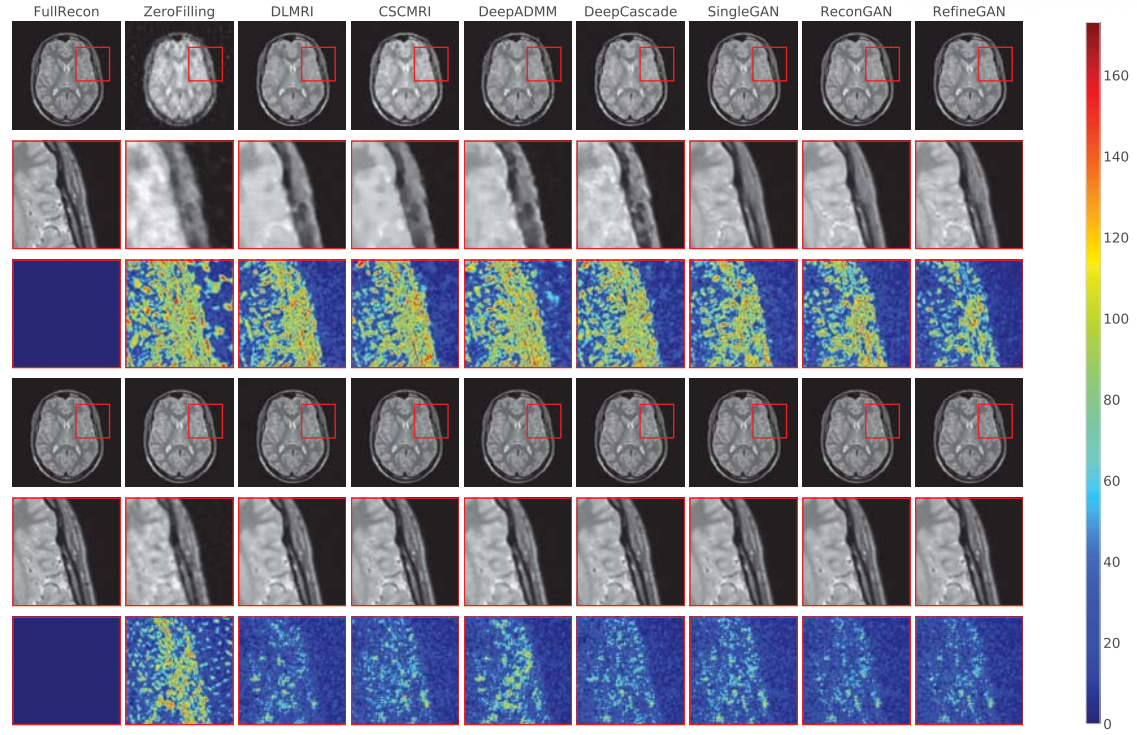
(e) Brain 30%

(f) Chest 30%

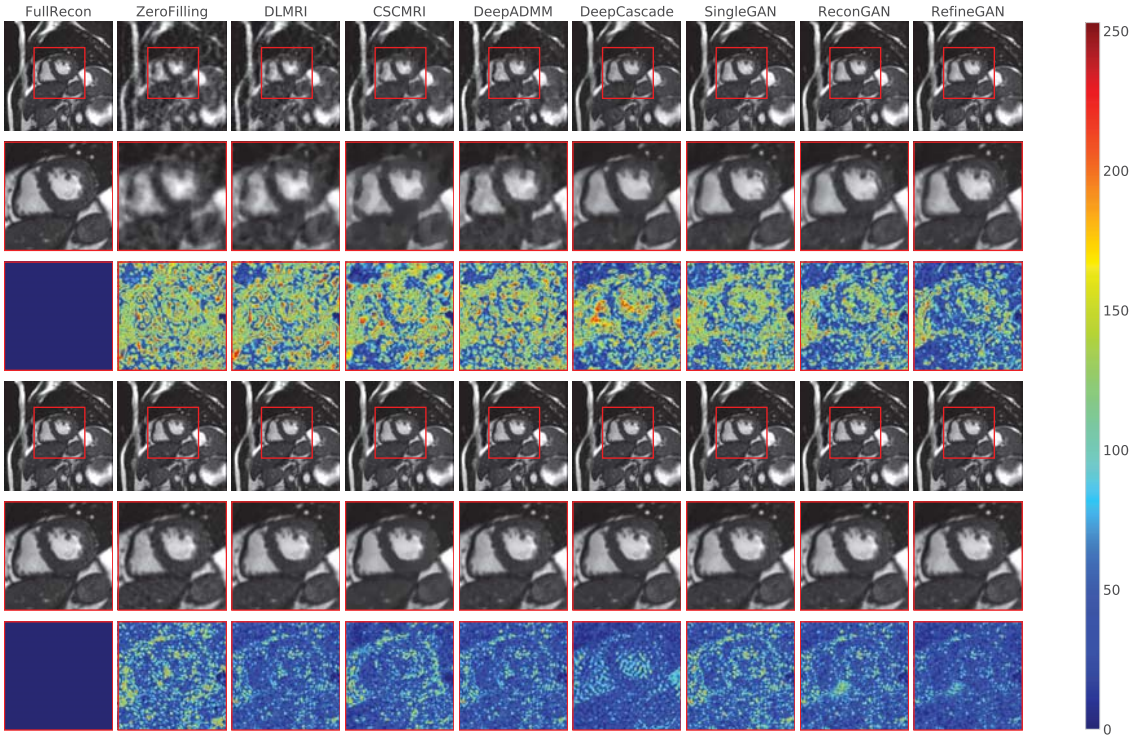
(g) Brain 40%

(h) Chest 40%

Figure 49: NRMSEs evaluation on the brain and chest test set



(a)



(b)

Figure 50: Image quality comparison on the brain (a) and chest dataset (b) at a sampling rate 10% (top 3 rows) and 30% (bottom 3 rows): Reconstruction image, zoom in result and 10× error map compared to the full reconstruction.

based method, DLMRI [18, 19, 104], solely relies on the CPU implementation of a greedy algorithm, so their reconstruction times are significantly longer (around 600 seconds) than those of the others with GPU-acceleration. Deep learning-based methods, including DeepADMM, DeepCascade, and our method, are extremely fast (e.g., less than a second) because deploying a feed-forward convolutional neural network is a single-pass image processing that can be accelerated using GPUs reasonably well. DeepADMM significantly accelerated time-consuming iterative computation to as low as 0.2 second. The running times of SingleGAN [91, 134] and our ReconGAN are, all similarly, about 0.07 second because they share the same network architecture (i.e., single-fold generator G). The running time of RefineGAN is about twice as long because two identical generators are serially chained in a single architecture, but it still runs at an interactive rate (around 0.1 second). As shown in this experiment, we observed that deep learning-based approaches are well-suited for CS-MRI in a time-critical clinical setup due to their extremely low running times.

Image Quality Evaluation: To assess the quality of reconstructed images, we use three image quality metrics, such as Peak-Signal-To-Noise ratio (PSNR), Structural Similarity (SSIM) and Normalized root-mean-square error (NRMSE) Figure 65, 48 and 66 show their PSNRs, SSIMs and NRMSEs error graphs, respectively. Additionally, Figure 50 shows the representative reconstruction of the brain and chest test sets, respectively, using various reconstruction methods at different sampling rates (10% and 30%) and their 10× magnified error plots using a jet color map (blue: low, red: high error). Overall, our methods (ReconGAN and RefineGAN) are able to reconstruct images with better PSNRs, SSIMs and NRMSEs. Note that we used the identical generator and discriminator networks (i.e., the same number of neurons) for SingleGAN, and our own method for a fair comparison. We observed that our cyclic loss increases the PSNR by around 1dB, and the refinement network further reduces the error to a similar degree. By qualitatively comparing the reconstructed results, we found that deep learning-based methods generate more natural images than dictionary-based methods. For example, CSCMRI and DLMRI produce cartoon-like piecewise linear images with sharp edges, which is mostly due to sparsity enforcement. In comparison, our method generates results that are much closer to full reconstructions while edges are still preserved; in addition, noise is significantly reduced. Note also that, comparing to the other CS-MRI methods, our method can generate superior results especially at extremely low sampling rates (as low as 10%, see Figure 50).

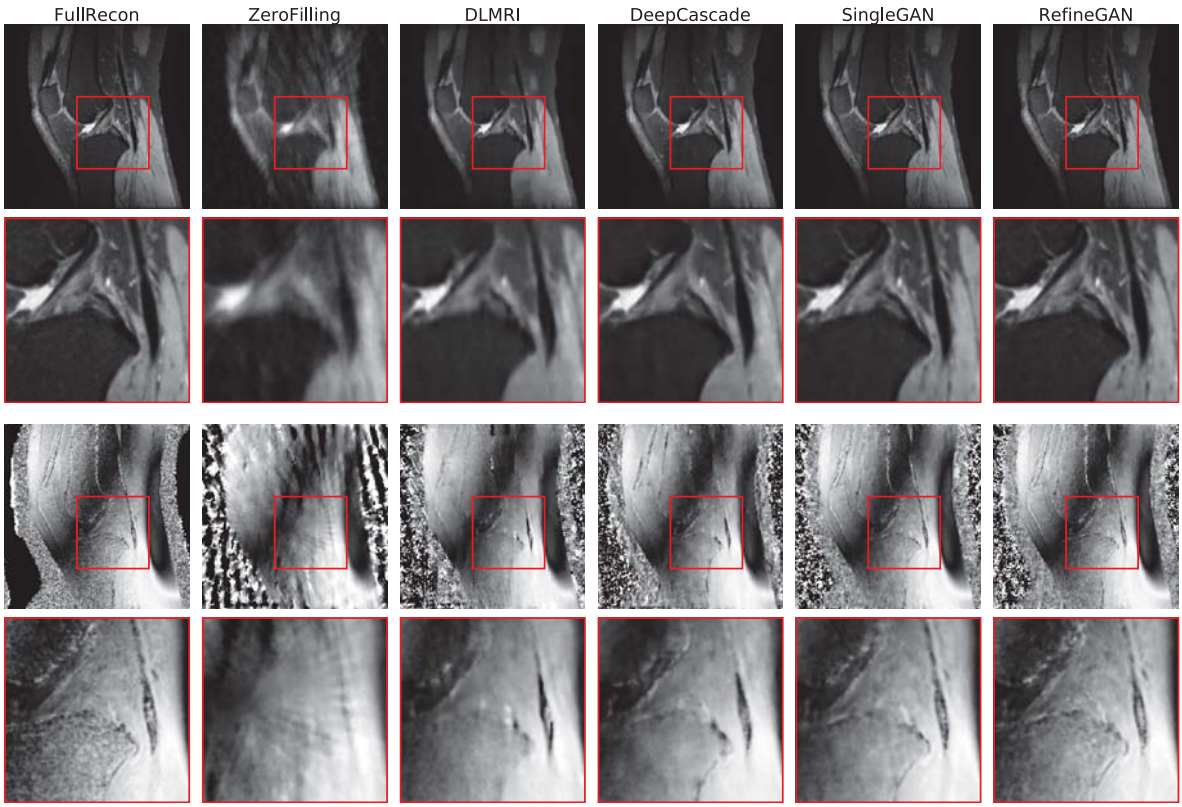


Figure 51: Image quality comparison on the knees dataset (top 2 rows: magnitudes, and bottom 2 rows: phases) at a sampling rate 10% : Reconstruction images and zoom-in results

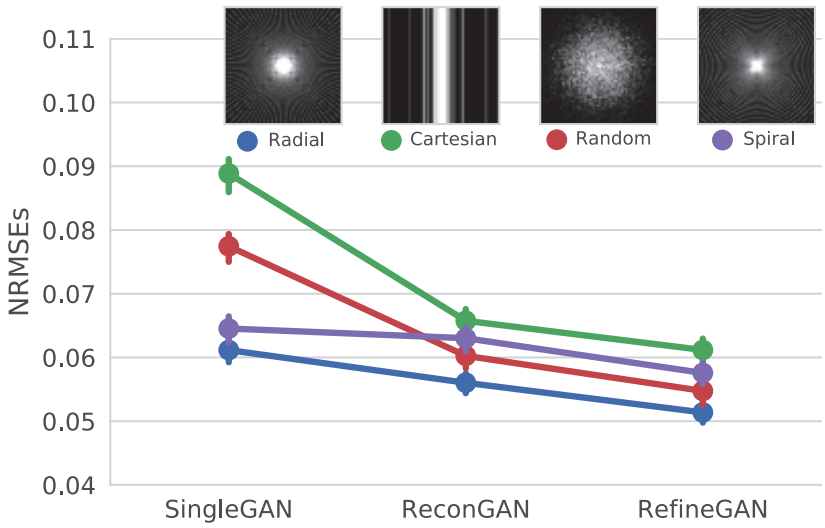


Figure 52: NRMSEs evaluation on the knees test set at sampling rate 20% with various masks

7.2.2 Results on complex-valued MRI data

The proposed method can accept 2-channel complex-valued zero-filling image as an input and return a 2-channel complex-valued reconstruction without loss of generality. We used another public database of MR k-space [16] (referred as the knees dataset) to evaluate our model. This opensource images consists of 20 cases of fully-sampled 3D Fast Spin Echo MR Images. We also chose randomly 10 slices in the middle of each case and further divided them into 2 sets: training and testing, 100 images each. Figure 51 depicts the representative reconstructions of knees test sets at the sampling rates 10% and their 10× magnified error plots on the image magnitude (top 3 rows) and phase (bottom 3 rows). As can be seen, the proposed RefineGAN can fruitfully reconstruct the result which has less error compared to other methods.

RefineGAN also consistently outperforms SingleGAN and ReconGAN for various sampling strategies. For example, Figure 52 visualizes the NRMSEs curves of the knees dataset using radial, cartesian, random and spiral sampling strategies (rate 20%). We observed that the radial sampling pattern results in the best performance among all. Moreover, our method reduces sampling-specific effects, i.e., difference between sampling strategies become less severe.

7.3 Summary

We introduced a novel deep learning-based generative adversarial model for solving the Compressed Sensing MRI reconstruction problem. The proposed architecture, RefineGAN, which is inspired by the most recent advanced neural networks, such as U-net, Residual CNN, and GANs, is specifically designed to have a deeper generator network G and is trained adversarially with the discriminator D with cyclic data consistency loss to promote better interpolation of the given undersampled k-space data for accurate end-to-end MR image reconstruction. We demonstrated that RefineGAN outperforms the state-of-the-art CS-MRI methods in terms of running time and image quality, thus indicating its usefulness for time-critical clinical applications.

In the future, we plan to conduct an in-depth analysis of RefineGAN to better understand the architecture, as well as constructing incredibly deep multi-fold chains with the hope of further improving reconstruction accuracy based on its target-driven characteristic. Extending RefineGAN to handle dynamic MRI is an immediate next research direction. Developing a distributed version of RefineGAN for parallel training and deployment on a cluster system is another research direction we wish to explore.

[16] http://mridata.org/fullysampled/knees

VIII Fully residual convolutional neural network for image segmentation in connectomics

8.1 Method

The motivation for the work presented here was the need for end-to-end neuron segmentation with higher accuracy. We observed, as discussed by others previously [57], that the current state-of-the-art U-net deep neural network [105] and conventional CNNs share a limitation in network depth due to gradient vanishing. To address this, we propose novel extensions of U-net that increase network depth through addition of residual layers in each network level with summation-based skip connections and incorporation of a chaining approach to further compensate for errors that arise in a single network. Our segmentation method achieves highly accurate results that outperform current state-of-the-art ssEM segmentation methods. The contributions of this work can be summarized as follows:

- We introduce *W-net*, a new end-to-end automatic ssEM image segmentation method using deep learning. Our solution is based on the combination of U-net and residual CNN properties with a novel chaining approach, resulting in an architecture that supports a self-correcting and self-refining model. Leveraging residual properties within and across levels in this fashion makes it possible to build a *deeper* network for achieving higher accuracy.

- We evaluate the performance of the W-net architecture by comparing it with current state-of-the-art methods listed in the leader board of the ISBI 2012 EM segmentation challenge. For segmentation accuracy, W-net outperformed all other top-ranked methods.

- We establish a *data enrichment* approach for ssEM data that collects all the orientation variants of input images. For the three-dimensional (3D) case, this yields sixteen total permutations via a combination of flipping, rotation, and re-ordering of adjacent slices (sections) in bins of three. The same process can be used during deployment, generating a final output that can contain many different probability values to enhance accuracy.

- We demonstrate the flexibility of W-net through two different ssEM segmentation tasks: cell nucleus segmentation and cellular membrane segmentation.

8.1.1 Data preparation

Data construction: To form a training instance, we leverage the 3D context in ssEM image stacks by grouping three consecutive slices including the current target and its adjacent neighbors. This makes it possible to resolve ambiguities in the z dimension (perpendicular to the ssEM sectioning plane). When the current target slice occupies the first or the last position in the stack, we use reverse optical flow with the target and existing neighbor image to estimate the empty third slice. The same strategy is applied when deploying for target segmentation. Note that this data construction method uses a two-dimensional (2D) convolution-based neural network, as opposed to the 3D version from Lee et al. [80].

Data enrichment: Distinct EM images frequently share similar orientation-independent textures associated with sub-cellular neuronal structures including mitochondria, axons, synapses, and more. We take advantage of this fact to enrich our training data for each slice by forming an additional fifteen other images and corresponding labels. For 2D enrichment, we generate eight permutations for each slice by rotating the original image by 90°, 180°, and 270° and then reflecting each case. Figure 53 shows the variants of these eight orientation changes. To take into account 3D information, we generate 3-slice bins for a every 2D orientation case with a sliding window along the z dimension. For each bin, the segmentation result is necessarily direction-independent from the top to bottom slice or the opposite. This enables us to further double the existing 8× enrichment by adding the reverse z ordering for each bin. Note that because the images and labels are enriched 16× in total, other on-the-fly data augmentation techniques—including random rotation, flipping, or transposition—are unnecessary with the exception of elastic deformation.

Elastic field deformation: To avoid an over-fitting case where the network remembers the training data, we perform per-instance elastic deformation over the entire enriched dataset for every training epoch. This strategy is common in machine learning, especially for deep networks, to overcome limitations of small training dataset sizes.

We first initialize a flow map consisting of a random sparse vector field (12×12) whose amplitudes at the boundaries have vanished (zero amplitude). This field is then interpolated to match the original image size and used to warp both image data and the corresponding labels. Figure 54 illustrates this procedure. The flow map is randomly generated so that deformations are independent in each training epoch.

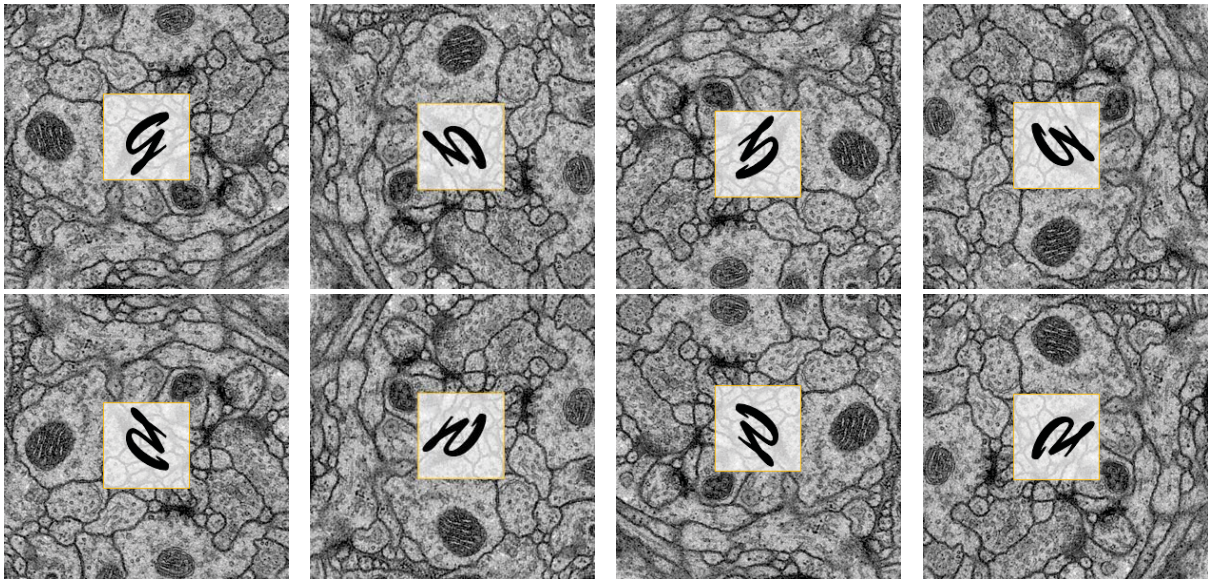
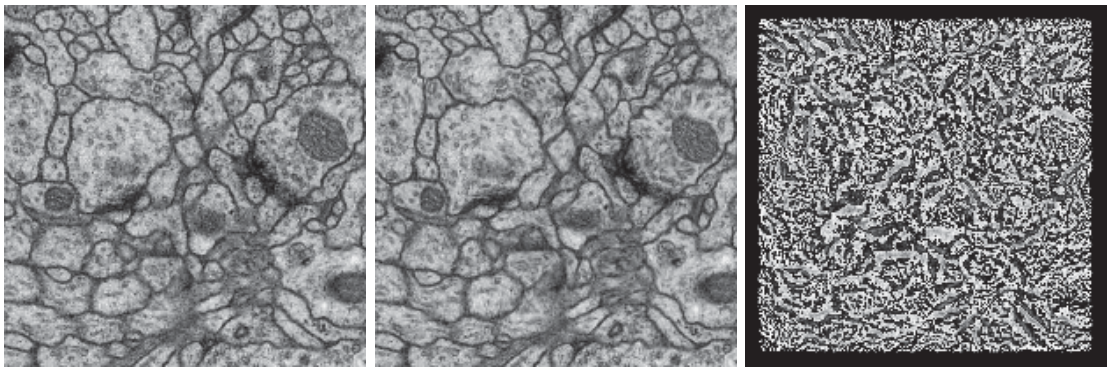
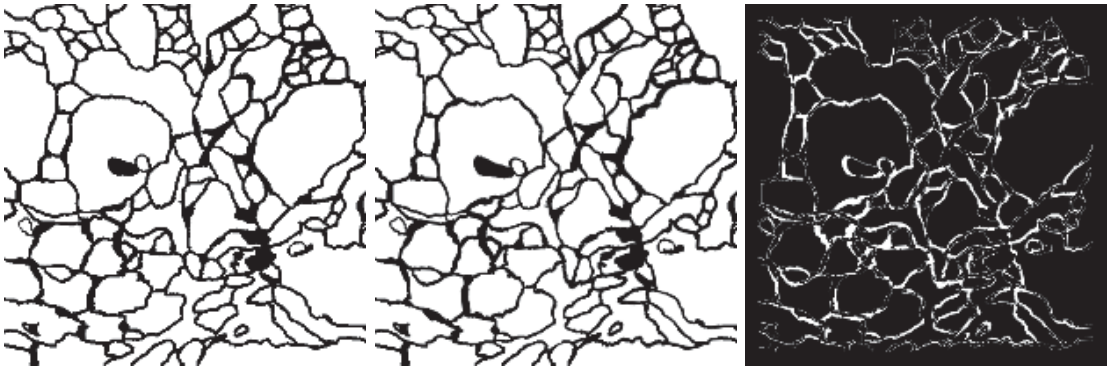


Figure 53: Data enrichment through these eight orientation variations would increase the input training data size by a factor of eight. The letter 'g' is overlaid for easier comparison.



(a) Before image warp (left), after warp (middle), and the difference (right).



(b) Before label warp (left), after warp (middle), and the difference (right).

Figure 54: Elastic field deformation example.

Random noise, boundary extension, random shuffle, data normalization and cross-validation: During the training phase, we randomly add Gaussian noise to the raw image (mean $\mu = 0$, variance $\sigma = 0.1$). Prior to the training phase, we perform a simple data normalization by scaling the image data in the range between $[0, 1]$. To this end, before fetching the enriched data for the model, we also shuffle the order of the dataset and perform a three-fold cross validation to improve the generalization of our method.

8.1.2 Chain of FusionBlocks: W-net architecture

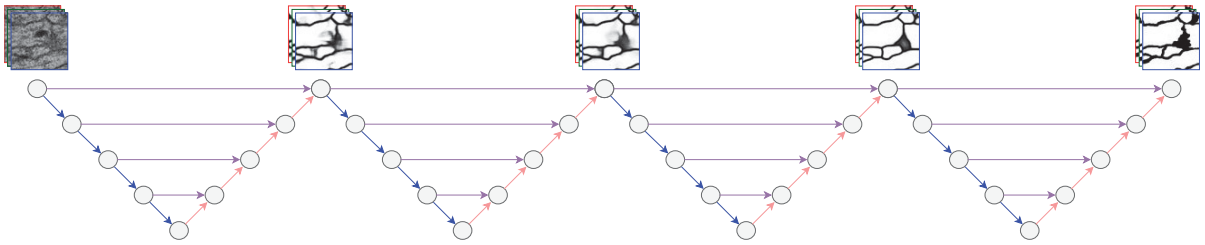


Figure 55: A chain of 4-fold concatenated FusionBlocks.

As can be seen, the proposed FusionBlock by itself, can perform an end-to-end segmentation from the input EM data to the final prediction of the segmentation. However, in the real world case, manual segmentation from human expert also takes another step to go through the detected labels and attempts to "correct" the mistakes. Therefore, we propose another step of refining the segmentation by concatenating a chain of several FusionBlocks to resolve the disambiguities of the first time predicted target probabilities. For example, figure 55 shows the 4-fold chaining FusionBlocks in a self-correcting form (W-net^4). By forcing the desired labels in between, the entire solution becomes target-driven approach. This enables our method to be a single-input and multi-output model, where each checkpoint in between attempts to produce better segmentation. Since the architecture of each individual FusionBlock is same, we can think of the proposed model is another variant of recurrent neural network which treats entire FusionBlock as a single state but the weights do not share after unfolding. The interesting point of view is FusionBlock can be considered as a V-cycle in the multi-grid method commonly used in numerical analysis, where the contraction in the encoding path is similar to *restriction* heads from fine to coarse grid, the expansion in the decoding path spans along the *prolongation* toward the final segmentation, and the skip connections play as the *relaxation*.

During the training, the weights of each FusionBlock are updated independently as opposed to the strategy of averaging the gradients from shared weights in Recurrent Neural Networks. Without loss of generality, let assume that we are training W-net^4 with the input images S and

their corresponding manual labels L. Each FusionBlock in W-net^4 is indexed as W-net$^4[i]$ where $i = 1, 2, 3$, and 4, and it generates the prediction $P[i]$ by minimizing the mean-absolute-error (MAE) loss between its prediction values and the target labels L. Per epoch, we incrementally train W-net$^4[i]$ and fix its weights before moving to the next W-net$^4[i+1]$. This procedure can be summarized as follows:

$$\min_{\text{W-net}^4[1]} \text{MAE}\left(P[1], L\right) \; s.t. \; P[1] = \text{W-net}^4\left[1\right](S) \tag{67}$$

$$\min_{\text{W-net}^4[2]} \text{MAE}\left(P[2], L\right) \; s.t. \; P[2] = \text{W-net}^4\left[2\right](P[1]) \tag{68}$$

$$\min_{\text{W-net}^4[3]} \text{MAE}\left(P[3], L\right) \; s.t. \; P[3] = \text{W-net}^4\left[3\right](P[2]) \tag{69}$$

$$\min_{\text{W-net}^4[4]} \text{MAE}\left(P[4], L\right) \; s.t. \; P[4] = \text{W-net}^4\left[4\right](P[3]) \tag{70}$$

The loss training curves decrease as i increases, and they are eventually converged as the number of training iterations (epoch) increases.

8.2 Result

8.2.1 Experimental Setup

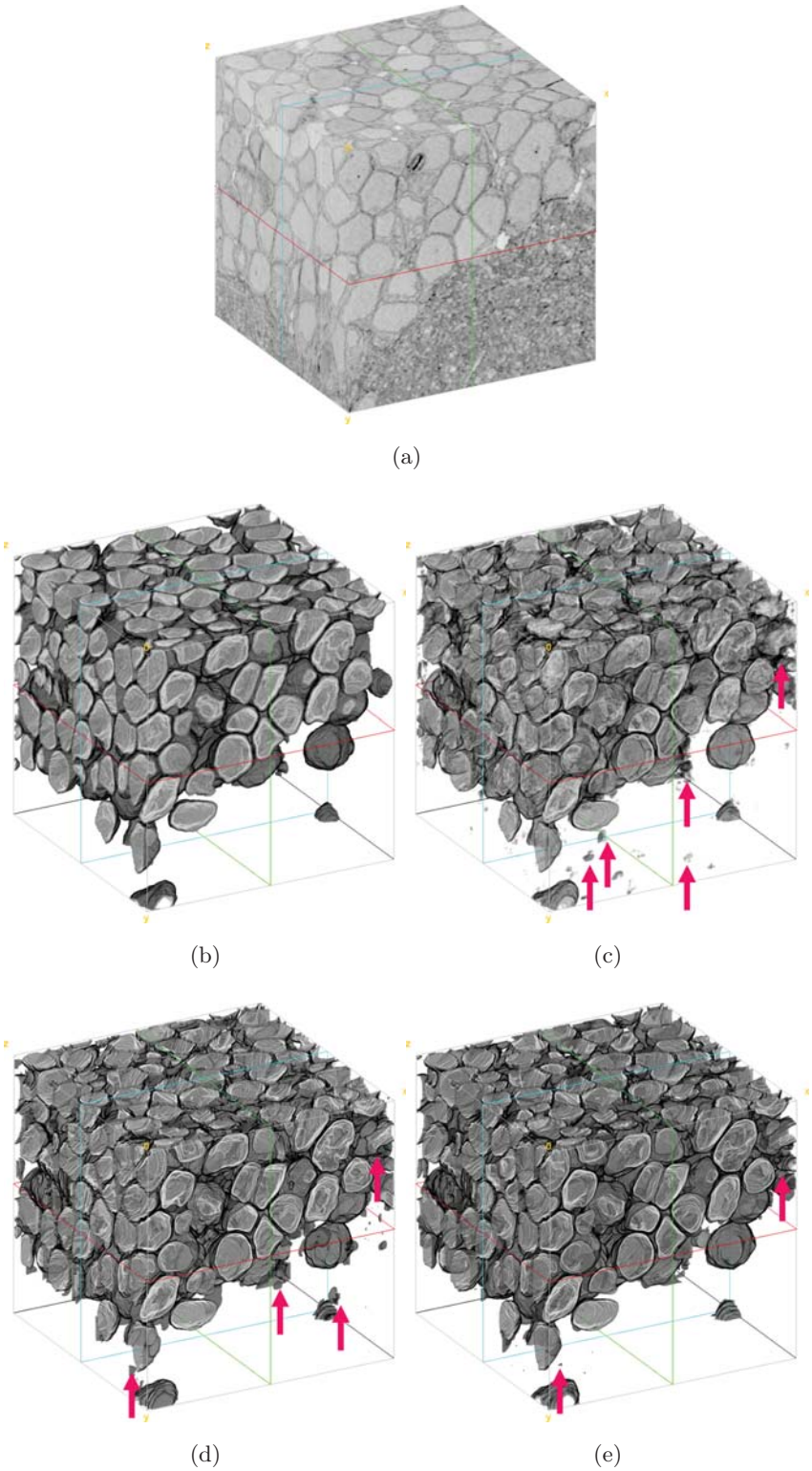
The proposed deep network is implemented using Keras open-source deep learning library [17]. This library provides an easy-to-use high-level programming API written in Python, where Theano or TensorFlow can be chosen for a backend deep learning engine. Training and deployment of the network is conducted on a PC equipped with an Intel i7 CPU with a 64 GB main memory and an NVIDIA GTX Geforce Titan X (Pascal) GPU. Since we use a data enrichment method that duplicates the input data by applying rotation and mirroring transformations for training, we apply the same transformations for deployment and average the results. We fixed all of the training parameters as the same across the experiments: using Nadam optimizer with learning rate is set to 0.0001 and the loss function is mean-absolute-error.

8.2.2 Larval zebrafish ssEM data

The larval zebrafish ssEM data analyzed here was taken from a publicly available database[18]. It was captured from a 5.5 days post-fertilization specimen. This specimen was cut into ∼18000 serial sections and collected onto a tape substrate with an ATUM [55]. A series of images spanning the anterior quarter of the larval zebrafish was next acquired at 56.4×56.4×∼60 nm^{3}vx^{-1}

[17] `http://keras.io`

[18] `http://zebrafish.link/hildebrand16/`



(a)

(b)

(c)

(d)

(e)

Figure 56: Visual comparison of the larval zebrafish ssEM volume segmentation; (a) input ssEM volume; (b) manual segmentation (ground truth); (c) U-net [105] result; (d) RDN [43] result; and (e) W-net$^{4}_{16}$ result. Red arrows are erroneous regions.

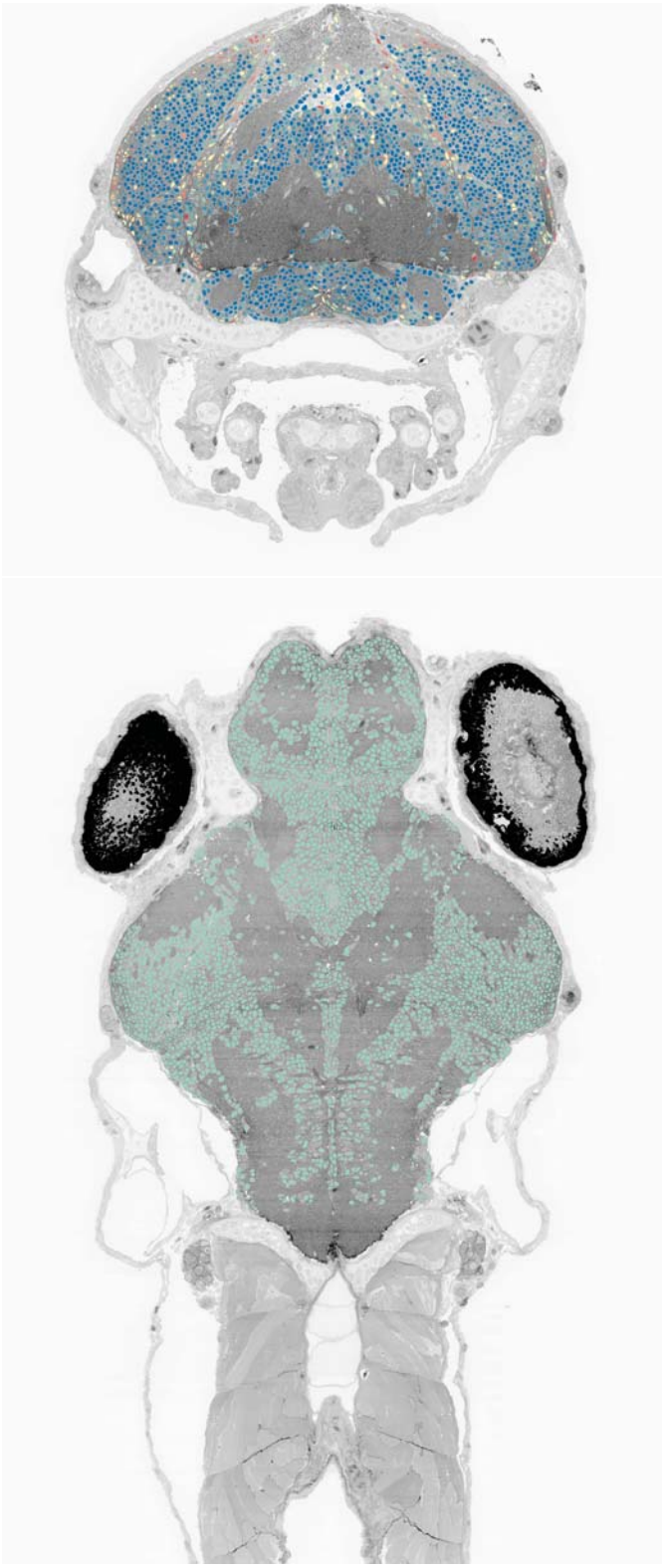


Figure 57: Cell nucleus segmentation results overlaid onto larval zebrafish ssEM dataset cross-sections through the transverse (top, blue to red color map for cell sphericity) and horizontal (bottom) planes.

Table 14: Segmentation accuracy on zebrafish ssEM dataset.

Methods	**W-net**	RDN [43]	U-net [105]
V_{rand}	**0.998648782**	0.991844302	0.987366177
V_{info}	**0.996929124**	0.994208722	0.992482059
V_{dice}	**0.963047248**	0.946099985	0.908491647

resolution from 16000 sections using scanning EM [60]. All images were then co-registered into a 3D volume using an FFT signal whitening approach [129].

For training, two small sub-region dataset crops were extracted from a near-final iteration of the full volume alignment in order to avoid ever deploying the segmentation on the training data. Two volumes were chosen to train on different tissue features. One volume was 512×512×512 and the other was 512×512×256. The features of interest—neuronal nuclei—were manually segmented as area-lists in each training volume using the Fiji [109] TrakEM2 plug-in [23]. These area-lists were exported as binary masks for use in the training procedure. For accuracy assessments, an additional non-overlapping 512×512×512 sub-volume test dataset was manually segmented. To differentiate various configurations of our method, we use the superscript to indicate how many FusionBlocks are inside W-net and the subscript to show the initial number of feature maps in the original resolution. For example, W-net$^4_{16}$ is the network that chains four FusionBlocks, each of them has $nb\_filters$ is equal to 16. Figure 56 displays volume renderings of the ssEM data, its manual nucleus segmentation, and segmentation results from U-net, RDN, and our model W-net$^4_{16}$. As shown, the proposed architecture introduced less false prediction compared to U-net and RDN (indicated by red arrows). Table 14 compares U-net and W-net$^4_{16}$ using three quality metrics, e.g., foreground-restricted Rand scoring after border thinning V_{rand} and foreground-restricted information theoretic scoring after border thinning V_{info} and the Dice coefficient V_{dice}, which also confirms that W-net$^4_{16}$ more accurate results than U-net [105] and RDN [43].

We deployed the trained network to the complete set of 16000 sections of the larval zebrafish brain imaged at 56.4×56.4×∼60 nm^{3}vx^{-1} resolution, which is around 1.2 terabytes in data size. We discovered that there are approximately 180,000 cell bodies in the larval zebrafish brain. Figure 57 shows ssEM dataset cross-sections in the transverse (top, x-y) and horizontal (bottom, x-z) planes of the larval zebrafish overlaid with cell nucleus segmentation. The transverse view also shows the sphericity (i.e., roundness) of each segmented cell nucleus in a blue to red color

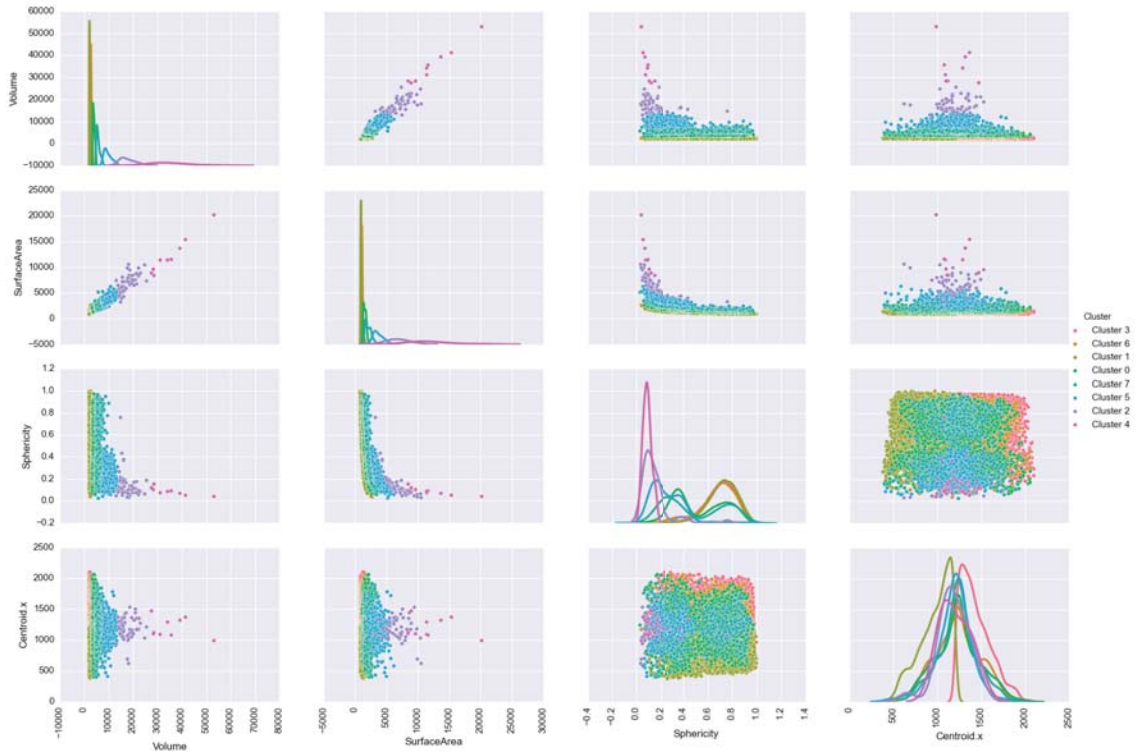


Figure 58: Correlation of statistical features: volume, surface area and sphericity, with horizontal coordinates of the centroids.

map, which helps to identify the location of non-nucleus false positives. We have conducted a statistical analysis of the cell morphological properties, such as volume, surface area, and sphericity. Those features are then clustered with a K-means algorithm and plotted in the form correlations, along side with their corresponding centroid coordinates. As illustrated in Figure 58, the outliers in the magenta cluster, correspond to cells with large volumes and surface areas, are clearly observed and can be eliminated by thresholding. We also observed that the feature distribution appears to be mirror symmetric across the midline of the larval zebrafish. This implies that the organization of neurons in the left and right brain hemisphere is likely to be very similar.

8.2.3 Larval Drosophila ssEM data

The *Drosophila* ventral nerve cord ssEM data analyzed here was captured from a first instar larva [22]. Training and test datasets were provided as part of the ISBI 2012 EM segmentation challenge [3]. Each was acquired at anisotropic $4\times4\times\sim50$ nm^3vx^{-1} resolution with transmission EM. The datasets were chosen in part because they contained noise and small image alignment errors that frequently occur in ssEM. For training, the provided set included a $2\times2\times1.5$ um^3

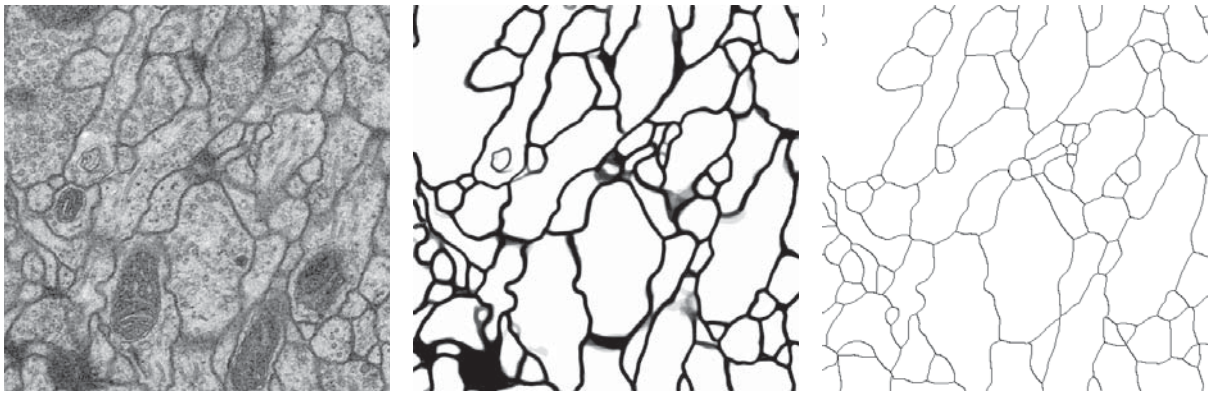


Figure 59: Example results of cellular membrane segmentation on test data from the ISBI 2012 EM segmentation challenge (slice 22/30) illustrating an input EM image (left), the probability prediction from our W-net$^2_{64}$ (middle), and the thinned probability prediction after applying LMC [6] post-processing (right).

Table 15: Accuracy of various segmentation methods on the *Drosophila* ssEM dataset (ISBI 2012 EM segmentation challenge leaderboard, March 2017)

Methods	V_{rand}	V_{info}
W-net$^2_{64}$ LMC	**0.983651122**	**0.991303595**
IAL MC/LMC [6]	0.982616131	0.989461939
IAL LMC [5]	0.982240005	0.988448278
W-net$^2_{64}$	**0.981586186**	**0.990099898**
PolyMtl [38]	0.980582825	0.988163049
KUnet [29]	0.980222514	0.988967601
FusionBlock (W-net$^1_{64}$)	**0.978042575**	**0.989945379**
IAL IC [84]	0.977345721	0.989240736
Masters [130]	0.977141154	0.987534429
CUMedVision [28]	0.976824580	0.988645822
ICNN [133]	0.976546913	0.988341665
DIVE-SCI [42]	0.976229111	0.987392123
LSTM [116]	0.975366444	0.987425430
U-net [105]	0.972760748	0.986616590

volume imaged from 30 sections and publicly available manual segmentations. For testing, the provided set included only image data, with segmentations kept private for the assessment of segmentation accuracy [3][19].

Figure 59 illustrates the results of our probability map extraction from test data without any post-processing step (middle) and with application of the lifted multicut (LMC) algorithm (right) [6], which resulted in its thinning. As depicted, our method, as with other state-of-the-art methods, is able to remove non-cellular membrane structures belonging to mitochondria (appearing as dark shaded textures) and vesicles (appearing as small circles). An uncertain region in the version without post-processing is illustrated by a gray region. In this case, the proposed network must decide whether the highlighted pixels should be segmented as membrane, but the region is ambiguous because of membrane smearing due to anisotropy in the data.

Our W-net approaches outperformed previous state-of-the-art methods on several standard metrics. These including foreground-restricted Rand scoring after border thinning (V_{rand}) and foreground-restricted information-theoretic scoring after border thinning (V_{info}), more detailed descriptions of which are available elsewhere [3]. Quantitative results for these metrics are summarized in Table 15. Even when a single FusionBlock module is used (W-net$^1_{64}$, $nb\_filter = 64$), we achieved better results compared to many well-known methods, such as U-net [105], the network-in-network approach [84], the fused-architecture approach [28], and the long-short term memory (LSTM) approach [116]. If we use a W-net with two FusionBlock modules (W-net$^2_{64}$, $nb\_filter = 64$), then our method outperformed the previous state-of-the-art deep learning methods without post-processing [29, 38]. These results confirm that chaining a deeper architecture with a residual bottleneck helps to increase the accuracy of the ssEM segmentation task. By applying the LMC post-processing to the result of W-net$^2_{64}$, our method ranked at the top in the ISBI 2012 EM segmentation challenge leaderboard (as of March 2017).

8.3 Discussion

Several recent work share related ideas with our W-net. Chen et al. [29] proposed concatenating multiple fully convolutional network to build RNN mainly for extracting inter-slice context. Unlike our W-net, Chen et al.'s approach feeds different resolutions of the input (multi-input) to produce the segmentation (single-output) for a single loss function. Wu proposed iteratively applying a pixel-wise CNN (ICNN) to refine membrane detection probability maps (MDPM) [133]. In this method, a regular CNN for generating MDPM from the raw input images and an iterative CNN for refining MDPM are trained independently, but our method trains as a single chained

[19] http://brainiac2.mit.edu/isbi_challenge/

network. In addition, our method can refine errors in MDPM better using a chained network (i.e., correcting errors in the error-corrected results), and scales better for a larger image size due to the end-to-end nature of the network.

Those related work and our proposed method try to leverage multiple input and output concurrently to improve accuracy. More in-depth analysis of why such approaches are beneficial to improve deep network's prediction accuracy is left for future work.

8.4 Summary

In this chapter, we introduced a novel deep network architecture for image segmentation that specifically targets ssEM image segmentation for connectomics analysis. The proposed architecture, W-net, was built by combining and extending U-net and residual CNN architectures to develop a deeper network for more accurate end-to-end image segmentation. We demonstrated the flexibility of W-net through its application to both cell nucleus (blob-type) and cellular membrane (membrane-type) segmentation tasks. In addition, we found that W-net achieved state-of-the-art performance in several standard quality metrics.

In the future, we plan to conduct an in-depth analysis of W-net to better understand the architecture and construct extremely deep W-net chains with the hope of further improving segmentation accuracy. We also intend to apply W-net to other connectomics segmentation tasks such as synapse detection. Developing a distributed version of W-net for parallel training and deployment on a cluster system is another research direction we wish to explore.

IX Deep Feature-aware Prior Denoising for Connectomics data

Electron microscopy (EM) is a crucial imaging technique in the neuroscience field of 'connectomics' research because only EM can provide sufficient image resolution to resolve densely packed neuronal structures that are nanometers in size [59, 83]. Such high-resolution imaging traditionally required human interaction for sample preparation and microscope operation, which served as bottlenecks in the data acquisition process. Recent advances in automatic tissue collecting and imaging techniques, such as the automated tape-collecting ultramicrotome (ATUM) [55] and the transmission electron microscope camera array (TEMCA) [11, 81, 121, 139], significantly reduce these acquisition burdens to make peta-scale data collection feasible. With the benefits of these high-throughput imaging techniques, however, also comes the introduction of various image artifacts that can make data analysis more challenging. For example, recent high-throughput transmission EM (TEM) techniques use a perforated tape covered with an electron-lucent support film to hold tissue samples and to convey them into the microscope for automatic and uninterrupted imaging (see Figure 60) [53, 62]. We observed that electron absorption property of some electron-lucent films is not spatially homogeneous, causing coherent noise as shown in Figure 61. Another example is electron beam charge damage artifacts, which are sometimes observed in the scanning EM (SEM). The thickness of tissue sections cut by an ATUM is typically about 30−50 nm, so electrons accumulated on the sample during imaging at high beam currents can cause permanent damage and leaves blob-like artifacts (see Figure 62). These artifacts are difficult to remove using conventional denoising filters developed for specific noise models.

We propose a novel semi-supervised learning-based denoising method that learns sources of image artifacts (i.e., noise) and removes them from unseen images. Unlike other deep learning-based noise removal methods that use supervised training to minimize the difference between the output of the network and the noise-free (clean) image, the proposed method uses noise examples selected from empty regions of EM images (that lack biological tissue) as a prior and constructs a network that can learn how to consume or extract that noise pattern. Our method consists of a three-way asymmetrically cyclic constraints in an adversarial deep network with two generators. One approximates the noise model (adding noise to a clean image), while the other is the inverse of the noise model (removing noise from a noisy image). The network is trained without paired ground truth data. This approach can avoid collecting ground truth clean and noisy image pairs, which are not feasible in many automatic imaging workflows, while providing a more general learning-based denoising model that can handle various unconventional EM imaging artifacts. The results show that the proposed method is more effective at removing

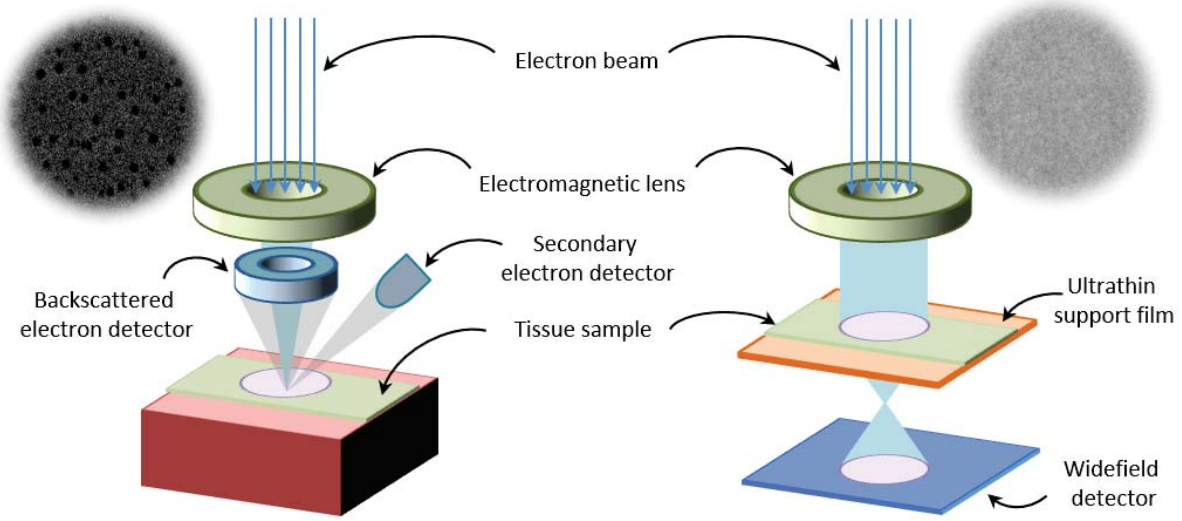


Figure 60: Schematic of SEM (left) and TEM (right) image acquisition workflows accompanied by examples of their artifacts (noise).

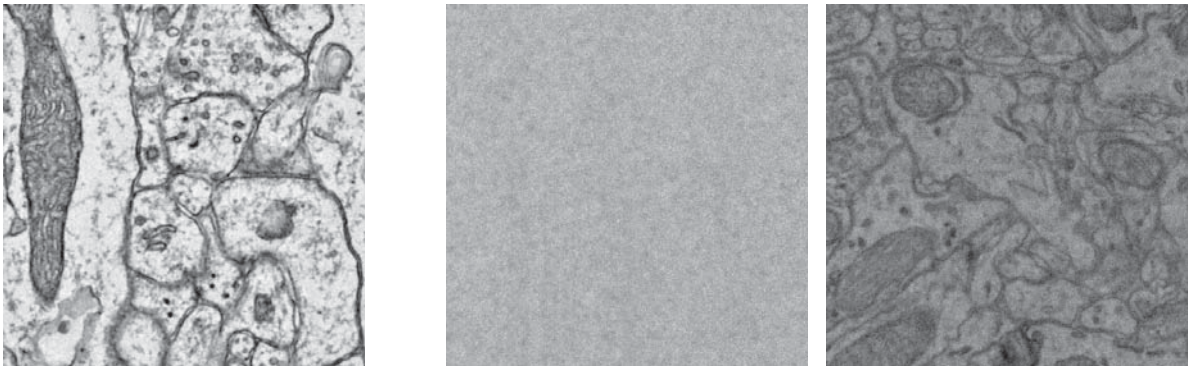
EM artifacts than current state-of-the-art denoising methods, including BM3D [34], dictionary-based [24], Noise2Noise [82], and CyclicGAN [140].

9.1 Method

9.1.1 Data preparation

Data for two example cases were prepared for our experiment; one is intra-EM type (i.e., same TEM for clean and noisy images) for film noise, and the other is inter-EM type (i.e., TEM for clean image and SEM for noisy image) for charge noise.

Selection of intra-type and Film noise samples: Thousands of thin sections of chemically fixed and stained mouse cortex tissue were collected onto two different support film substrates compatible with TEM: pioloform or LUXfilm®. The TEM_{DR5} dataset was acquired from mouse visual cortex tissue sections collected onto pioloform support film. The TEM_{PPC} dataset was acquired from mouse posterior parietal cortex tissue sections collected onto LUXfilm® support film (Luxel Corporation). Pioloform supports films are thinner and more fragile, but contribute little noise to the images. LUXfilm® support films are more robust and better suited for automatic sectioning workflows [53] but can add substantial noise to the images. TEM images were acquired at $4.3\times4.3\times \sim 40$ $\text{nm}^3\text{vx}^{-1}$ resolution using a modified JEOL 1200CX system. To model the film noise, we also captured images of LUXfilm® support films lacking tissue sections.



(a) TEM_{DR5} (b) Film noise (c) TEM_{PPC}

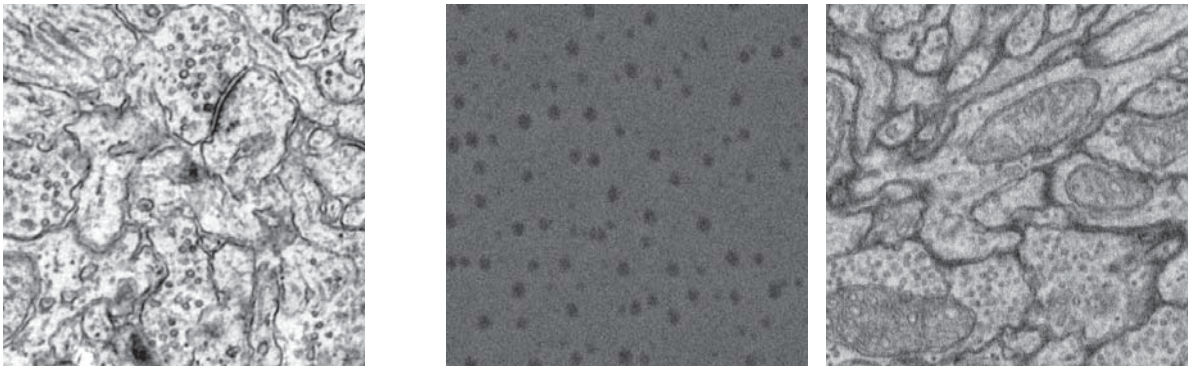
Figure 61: Example of intra-type and film noise images. (a) is a clean TEM image, (b) is a film noise image, and (c) is a noisy TEM image.

By superimposing film noise onto clean images (from TEM_{DR5}) via pixel-wise multiplication, we created synthetic noisy images for validating the trained network. These synthetic noise images were not used for training our model. Figure 61 shows examples of typical TEM_{DR5}, Film noise, and TEM_{PPC} images.

Selection of inter-type and Charge noise samples: Images were acquired from 5−7 days post-fertilization larval zebrafish brain tissue with both TEM and SEM methods. The TEM_{ZB} images were captured at a resolution of $4.0\times4.0\times \sim 40$ $\text{nm}^3\text{vx}^{-1}$ using a modified JEOL 1200CX TEM system. The SEM_{ZB} images were captured at a resolution of $4.0 \times 4.0 \times \sim 60$ $\text{nm}^3\text{vx}^{-1}$ using a FEI Magellan XHR 400L SEM system [61]. Note that these patches were extracted randomly with the given criteria in the collected image section. Figure 62 shows examples of typical clean TEM_{ZB} images and SEM_{ZB} noise patterns and noisy tiles. In addition, the tissue samples in this selection are different from above (zebrafish versus fly). We abbreviate these indicators in the subscriptions of the dataset names (DR5, PPC, ZB).

9.1.2 Overview of the proposed method

Figure 63 describes the architecture overview of the proposed denoising method. The input to our method is a collection of three types of images; clean EM images (such as TEM_{ZB} or TEM_{DR5}), pure noise images cut out from the empty region of noisy EM images (film or charging noise patterns), and noisy EM images (such as SEM_{ZB} or TEM_{PPC}). Note that those three input images are acquired independently and we do not use paired data to train the proposed network.



(a) TEM_{ZB} (b) Charge noise (c) SEM_{ZB}

Figure 62: Example of inter-type and charge noise images. (a) is a clean TEM image, (b) is a charge noise image, and (c) is a noisy SEM image.

The generator $\mathbf{G}$ in our model consists of two end-to-end convolutional-autoencoder networks: $\mathbf{G_1}$ synthesizes a *noisy-like* image I_f from a 2-channel image, i.e., a merge between the clean image C_t and the noise pattern N_t that were sampled randomly from big tiles of truly clean image and noise motif training datasets; $\mathbf{G_2}$ decomposes the input noisy image I_t, sampled randomly from real noisy EM training images, into a 2-channel image, which is a concatenation of the *clean-like* image C_f and the *noise-like* image N_f. Mathematically, the above descriptions can be formalized defined as follows, in which $\mathbf{G_1}$ and $\mathbf{G_2}$ are considered being universal approximate functions:

$$\begin{aligned} I_f &= \mathbf{G_1}(C_t, N_t) \\ C_f, N_f &= \mathbf{G_2}(I_t) \end{aligned} \tag{71}$$

$$\begin{aligned} C_r, N_r &= \mathbf{G_2}(I_f) \\ I_r &= \mathbf{G_1}(C_f, N_f) \end{aligned} \tag{72}$$

There are two different paths for training; The upper direction encourages the reconstruction of input clean image and noise pattern after going through $\mathbf{G_1}$ and $\mathbf{G_2}$. The lower path acts similarly but in reverse order: it attempts to reassemble the input noisy image by subsequently proceeding over $\mathbf{G_2}$ and $\mathbf{G_1}$. This serves as strong constraints for three independently generated input images.

The discriminators $\mathbf{D}$ including $\mathbf{D_I}$, $\mathbf{D_C}$, $\mathbf{D_N}$ for recognizing clean images C, noise patterns N and noisy images I, respectively, attempt to differentiate between the real instances from the database and the fake results output generated by $\mathbf{G}$. It is worth noting that although $\mathbf{G_1}$, $\mathbf{G_2}$ are sharing the similar architecture of a convolutional-autoencoder and the same structure of such a binary classifier is used to construct $\mathbf{D_C}$, $\mathbf{D_N}$, $\mathbf{D_I}$, each of them has its own variable

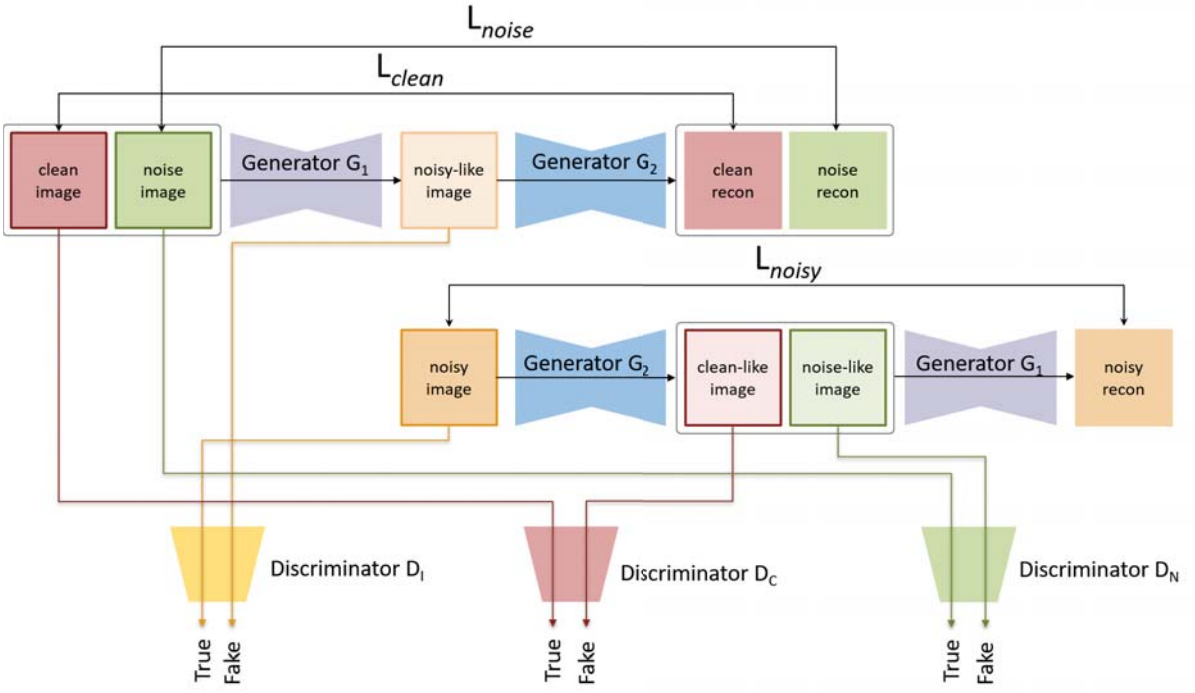


Figure 63: The architecture of the proposed model

scope and hence will be updated differently during the training. The inner configurations, such as number of filters, residual bottlenecks, etc., are made identical to CycleGAN [140] and therefore allow to conduct direct comparisons afterward. The entire system involves training **G**, **D** adversarially until a balance is reached at the convergence stage.

9.1.3 Loss definition

We wish to train **D** so that it can maximize the probability of assigning the correct *true* or *false* label to clean, noise and noisy images. Note that the objective function for **D** can be interpreted as maximizing the log-likelihood for estimating the conditional probability, where the image comes from: $\mathbf{D_I}(I_t) = \mathbf{D_I}(\mathbf{G_1}(C_t, N_t)) = 0$ $(fake)$, and $\mathbf{D_I}(I) = 1$ $(real)$. Simultaneously, generator $\mathbf{G_1}$ is trained to minimize $[1 - \log \mathbf{D_I}(I_t)]$ or $[1 - \log \mathbf{D_I}(\mathbf{G_1}(C_t, N_t))]$. This can be addressed by formally defining an adversarial loss L_{adv}:

$$\begin{aligned} L_{adv} = [1 - \log \mathbf{D_C}(C_f)] \quad &+ [\log \mathbf{D_C}(C_t)] \\ + [1 - \log \mathbf{D_N}(N_f)] \quad &+ [\log \mathbf{D_N}(N_t)] \\ + [1 - \log \mathbf{D_I}(I_f)] \quad &+ [\log \mathbf{D_I}(I_t)] \end{aligned} \tag{73}$$

In an extreme case, with large enough resources and data, the network can overkill the significant feature on reconstruction C_f. To avoid this case, we introduce an additional constraint,

the data consistency loss L_{cyc}, which is a combination of each input image L_{clean}, L_{noise} and L_{noisy} in a cyclic fashion. This mean that the noise pattern can be consolidated with the clean image to make the noisy-like image and separated thereafter in order to satisfy the conservative energy consumption. In practice, various distance metrics, such as mean-square-error (MSE), mean-absolute-error (MAE), etc, can be employed to implement L_{cyc}. Note that the cyclic loss only affects the generator **G** and not the discriminator **D**.

$$L_{clean} = \mathbf{d}(C_t, C_r), \quad L_{noise} = \mathbf{d}(N_t, N_r), \quad L_{image} = \mathbf{d}(I_t, I_r) \tag{74}$$

$$L_{cyc} = L_{clean} + L_{noise} + L_{image} \tag{75}$$

In summary, our system involves two sub-networks which are trained adversarially to minimize the following loss:

$$L_{total} = L_{adv} + \lambda L_{cyc} \tag{76}$$

where λ is used to control the balance between each contribution. We set $\lambda = 10$ for all the experiments. The Adam optimizer [76] is used with the initial learning rate of $1e^{-4}$, which decreases monotonically over 500 epochs. The entire framework was implemented using a system-oriented programming wrapper (tensorpack [20]) of the tensorflow [21] library.

9.2 Training and Testing specification

Training phase: The proposed model accepts a certain sizes of Field of View (FoV) as 512×512 meanwhile the actual collected images have much larger size. For each kind of dataset, i.e., clean, noise and noisy images, we randomly sample an image patch that has exactly the size of 512×512 to train the proposed GAN method. Due to this strategy, highly likely that we can avoid overfitting case since the training instances are always renewed for each iteration. The Adam optimizer has been employed with the initial learning rate is set at $2e^{-4}$. We train our model in 2000 epochs and periodically save the checkpoints.

Testing phase: To deploy the trained model on the test noisy image and obtain the denoised tile, we subdivide it into many overlapped patches which have the same FoV 512×512 and the step stride is 256. The prediction of each patch is then multiplied by a Gaussian weight and the result is obtained by dividing weighted estimation to the total per-pixel weight. Figure 64 shown an example of the final weight maps on a typical noisy image 2048×2048. By doing this, we can effectively avoid the window artifact from a naive subdivision approach with or without the halo extensions.

[20] `http://tensorpack.readthedocs.io/`
[21] `http://www.tensorflow.org/`

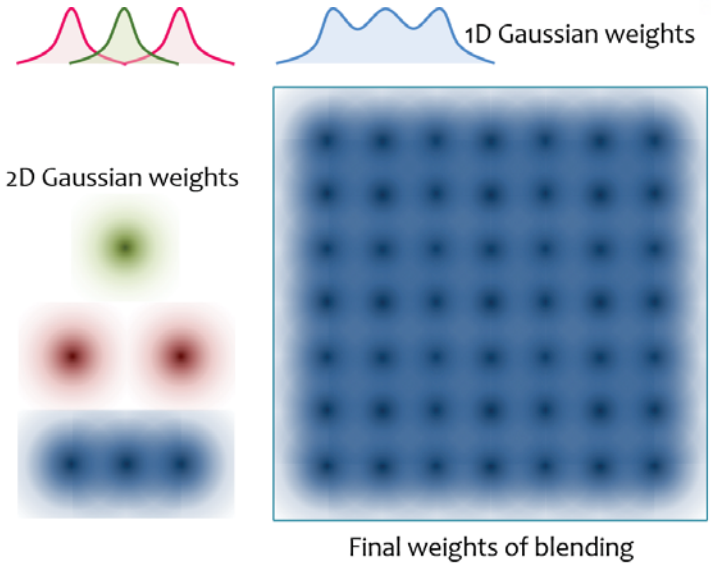


Figure 64: Fully weighted map of overlapping 512^2 blocks on a 2048^2 image with a stride 256

9.3 Result

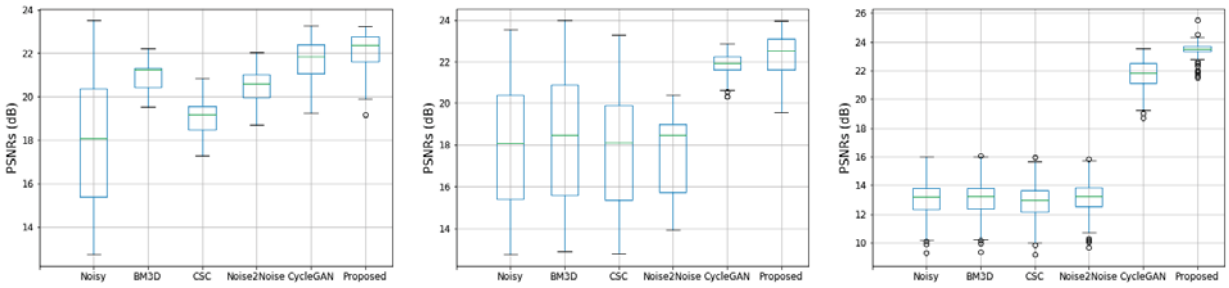
9.3.1 Experiment setup

Table 16: Specifications of our cases for examination

	Clean Images	Noise types	Noisy Images (types)
1	TEM_{ZB}	Gaussian	TEM_{ZB} + Gaussian (Synthetic)
2	TEM_{ZB}	Charge	TEM_{ZB} + Charge (Synthetic)
3	N/A	Charge	SEM_{ZB} (Real)
4	TEM_{DR5}	Film	TEM_{DR5} * Film (Synthetic)
5	N/A	Film	TEM_{PPC} (Real)

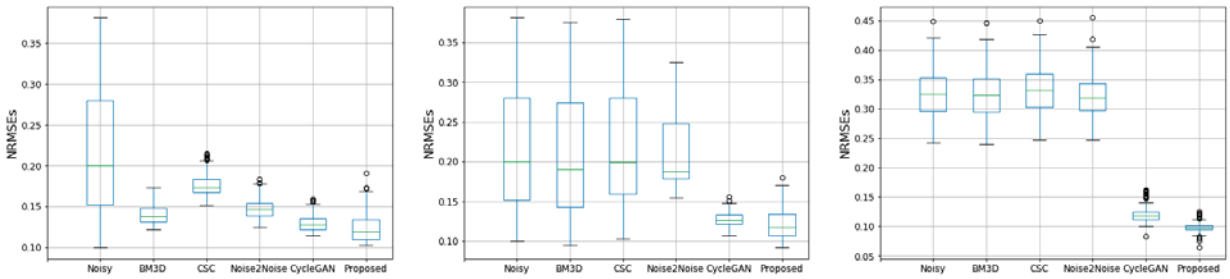
Table 16 summarizes the experiments we conducted to assess the performance of our method. There are three types of noise, i.e., Gaussian noise, charge noise, and film noise. Even though Gaussian noise is not the main target of our denoising method, it serves as a standard metric to compare with other existing denoising methods specifically designed for that noise model.

We conducted both quantitative and qualitative evaluations. For quantitative evaluations (cases 1, 2, and 4), we generated synthetic noisy images by adding (or multiplying) either synthetic Gaussian noise ($\mu = 0.0$, $\sigma = 0.05$) or real charge and film noise patterns to clean TEM



(a) Case 1 (b) Case 2 (c) Case 4

Figure 65: PSNRs (in dB) of related methods compared to the ground truth images.



(a) Case 1 (b) Case 2 (c) Case 4

Figure 66: NRMSEs of related methods compared to the ground truth images.

(TEM_{ZB} and TEM_{DR5}), applied denoising methods (ours, BM3D [34], CSC [24], Noise2Noise [82], and CycleGAN [140]), and compared the results with the ground truth clean images using the Peak-Signal-to-Noise-Ratio (PSNR) (Figure 65) and the Normalized Root Mean square Error (NRMSE) (Figure 66) metrics. For cases 3 and 5, we do not have ground truth clean images, so we made visual comparisons between denoised results. We used 128 test images for each experiment.

9.3.2 Quantitative and qualitative evaluations

Case 1 (TEM_{ZB} and Gaussian noise):

As we can see, results from case 1 (Gaussian additive noise, Figure 65a and Figure 66a) show that performance of CSC [24] is less susceptible to recover the noise-free inputs. Among all, since our approach inherits both benefits from prior distribution (noise model assumed to be known as Gaussian) and hard complementary distribution (input noise should be able to regain), it outperforms Noise2Noise [82], and CycleGAN [140] by a little margins. As results, the differences between the reconstruction and the ground truth from our method are less severe compared to others (see Figure 67). Hence, we conclude that training a denoiser with hard constraint to recover the input noise delivers competitive image quality under the assumption of incoherent noise-type like additive Gaussian.

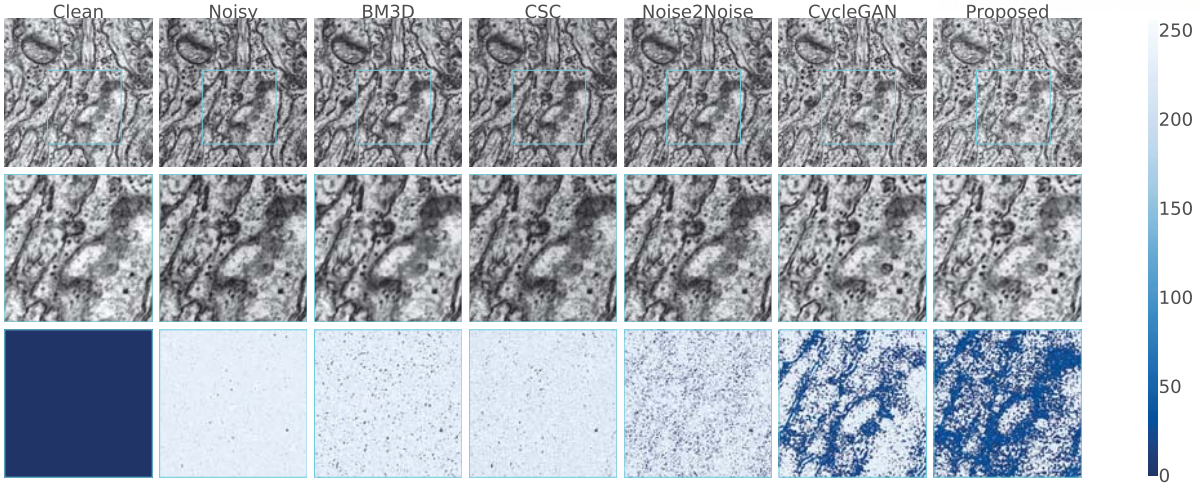


Figure 67: Comparison on TEM_{ZB} dataset in case 1.

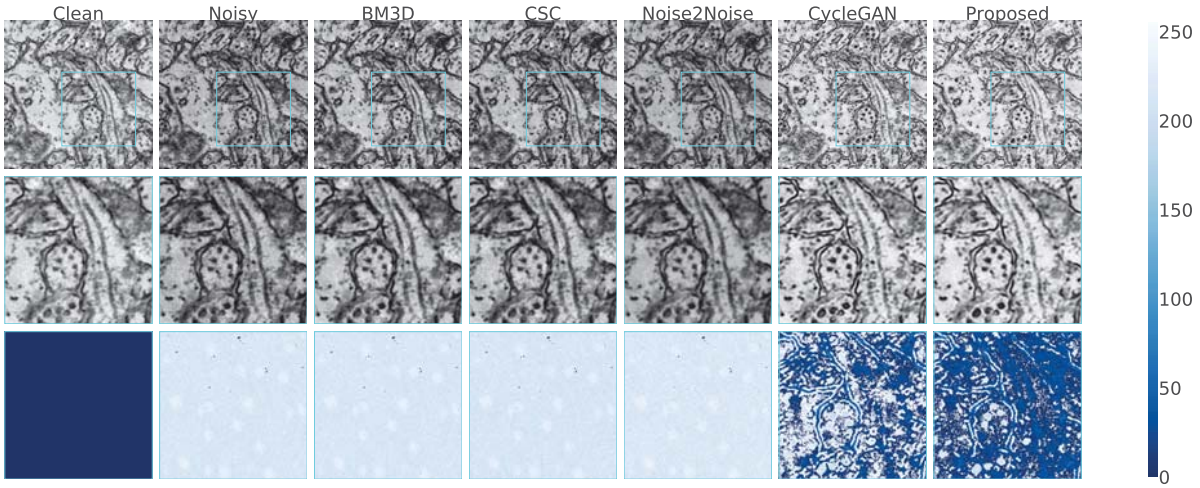


Figure 68: Comparison on TEM_{ZB} dataset in case 2.

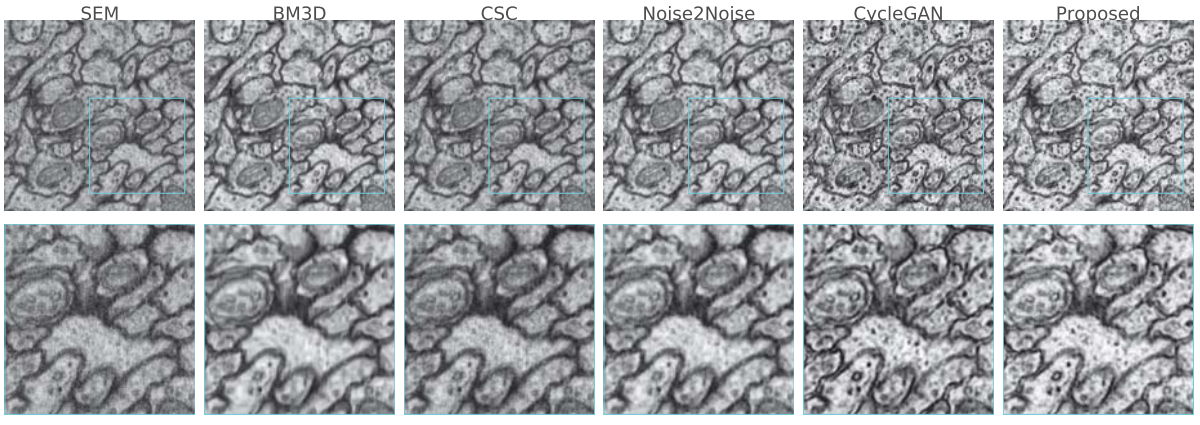


Figure 69: Comparison on SEM_{ZB} dataset in case 3.

Case 2 (TEM_{ZB} and Charge noise): By switching from Gaussian to Charge noise (case 2, Figure 65b and Figure 66b), although the artifact-interference strategy is still addition as before, it is not easy to achieve the PSNRs > 22 dB except CycleGAN [140] and the proposed method. Figure 68 pictorially confirms that the coherence electric charge damage noise caused by the photon-beam is effectively removed with the presence of cyclic loss in which our model elevates for. This type of coherent noise has blob-like artifacts on top of the Gaussian pattern scattered across the image and cannot be well-defined by mathematical formulas, which leads to the fact that designing such a series of hand-crafted filters is not easy. Therefore, approximating the forward and inverse noise models by neural networks has certain benefits in which locations and sizes of electric charge damages can be incorporated as conditions for the learning process.

Case 3 (SEM_{ZB} and Charge noise): We used the model trained with TEM_{ZB} data and Charge noise to blindly deploy on unseen SEM_{ZB} images. Figure 69 illustrates side-by-side the comparison of results from mentioned methods including ours. We can visually assess that CycleGAN and our method can effectively remove background blob-like charge noise as well as increase signal-to-noise of SEM images because our discriminators are trained using clean TEM images. One can also find subtle differences between ours and CycleGAN; ours tends to reconstruct background clearer and edges are sharper than CycleGAN, which could be due to the separate discriminators for clean and noise images in the proposed method.

Case 4 (TEM_{DR5} and Film noise): Similar achievements are obtained if the multiplicative noise is used to assess the reconstruction performance in case 4. Figure 65c and Figure 66c show that the proposed method is not only superior across image modalities (TEM_{ZB} versus SEM_{ZB} in case 2 and 3) but also within the same scope of imaging method aside from different image types and noise behaviors, i.e., TEM_{DR5} and TEM_{PPC} and Film noise. Significant improvements from CycleGAN [140] and our model over the rest indicate that the noise-like photon-absorption artifacts of the film can not be estimated via any noise assumptions, and therefore cyclically reconstructing the noise-altered and noise-free images can maintain the DC components (i.e., low frequency intensities) of the data. Overall, as shown in Figure 70, the proposed method generates less errors compared to the ground truth than the others.

Case 5 (TEM_{PPC} and Film noise): Figure 71 visually compares the results of ours and the others. Overall, the results seem similar to case 3, except that TEM_{PPC} is in general darker because the film noise suppresses the contrast of the image and our method and CycleGAN restored the contrast closer to that of clean TEM images. It is clearly shown that ours and CycleGAN outperform the rest, and ours is less blurry than CycleGAN (especially, vesicles and synaptic clefts are clearer in our result).

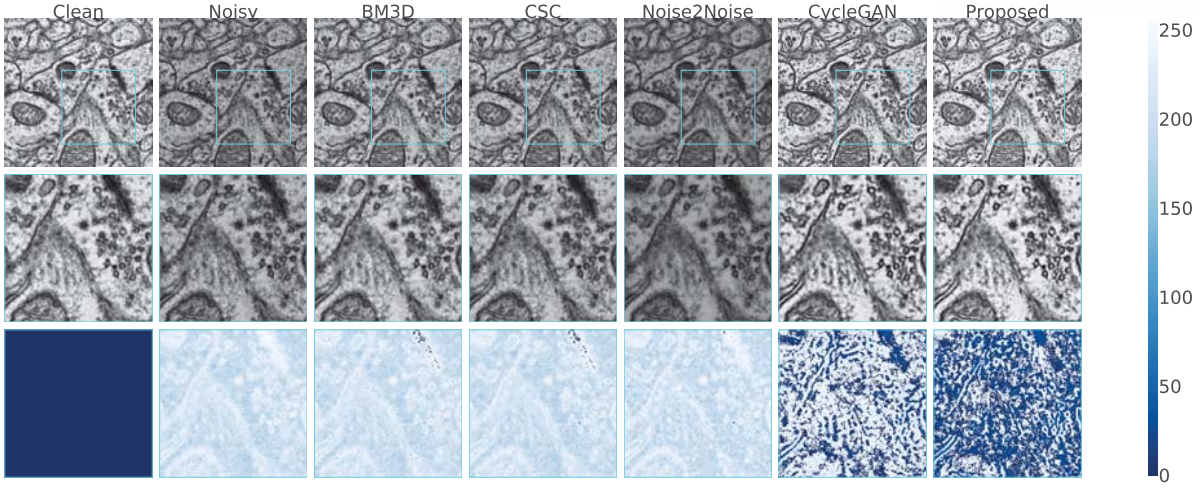


Figure 70: Comparison on TEM_{DR5} dataset in case 4.

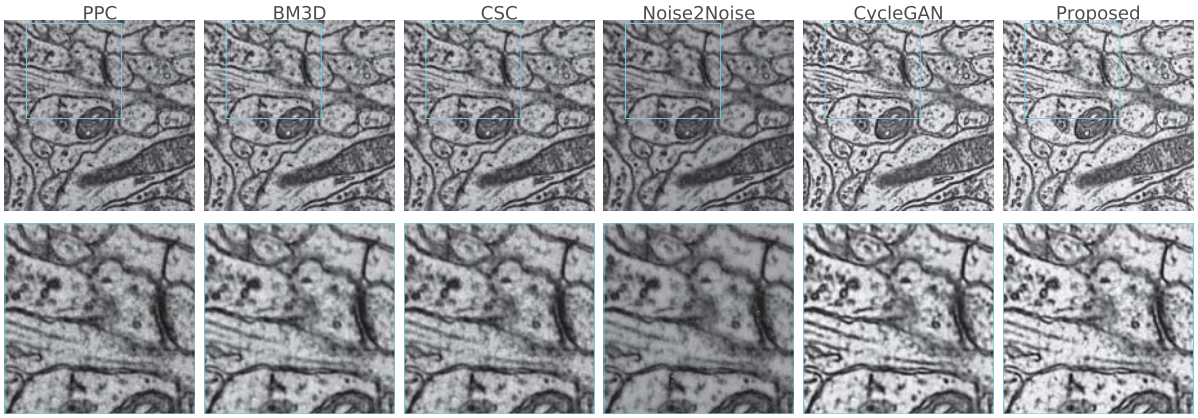


Figure 71: Comparison on TEM_{PPC} dataset in case 5.

9.4 Summary

In this work, an asymmetrically cyclic adversarial network is proposed to perform denoising tasks, specifically designed for Electron Microscopic (EM) analysis pipeline. Unlike other common deep learning methods that have been trained with pairs in which the target instances are usually noise-free images, our work focuses on empirical observations where the noise-only patterns (on films or electric charge distribution) can be estimated during the imaging process, and uses those as priors, along side with generative adversarial loss, to consolidate noisy-like images and delineate to reconstruct the original noise. We demonstrated that the proposed method outperform CycleGAN which is an unpaired image-to-image translation method, and other state-of-the-art deep learning-based approaches on the unseen images which have coherent distributions. Thanks to the nature of one-time feedforward deployment on trained neural networks, it will have a potential to be applied to a massive number of image sections generated in high-throughput EM imaging workflow.

X Conclusion and Future Work

10.1 Summary of Dissertation Research

In this dissertation, I have presented several approaches to tackle and solve diverse biomedical image reconstruction tasks including restoration/enhancement, segmentation and denoising. Recently, however, new methods for measuring the activity and determining the structure of neural circuits developed by a diverse group of scientists is enabling unprecedented exploration of how the most complicated organ in our body generates our complex behaviors. During a joint work with Harvard Medical School, I came to respect the highly complex nature of neural data, the analysis of which requires combining the knowledge of wide range interdisciplinary research fields such as biology, engineering, and many other natural sciences. The data collected from state-of-the-art neuroscience techniques (e.g., serial-section EM, ECG, EEG) are massive and very high-dimensional in general. From this work, I have come to appreciate that the key to successful exploration of neural data is the use of computationally efficient and interactive design tools for scientific computing and visualization. This is where I see big potential for computer science to enhance neuroscience investigations through the use of many-core GPU accelerators.

In our modern world these days, seeing or visualizing data gains much more attentions for clarifying the theory of domain specific knowledge. For instance, in connectomics, with the help of Electron Microscope, scientists are able to discover the very complex structures such as neurons and their corresponding axons, synapses, vesicles and so on. Therefore, they can construct experimental proofs that support or protest our ancients' classical observations with an absence of nano-scaled instruments.

The term connectomics stands for a brand-new and cutting-edge research direction in basic sciences, for more specific, neuroscience. People who work in this field are particularly interested in how our human brains work and wish to understand the way that our memory is stored and processed. They started by investigating the worm C. elegans which has a modest neural system including around 7000 connections between roughly 300 neurons (in 1970s), then subsequently approached to bigger species like zebra fishes, mice, chimpanzees, etc. but rarely to human. At this moment, with the stage of the art medical equipment explosion, we are about to enter the era of revealing our most complex organ's behaviors: the brain.

Research ideas for using GPGPU to solve problems in biomedical image reconstruction are numerous, and my research so far has just scraped the surface of these possibilities. To this end, I believe that this dissertation would provide a perfect place to exchange my PhD research on the use of many-core accelerator GPUs to solve problems in biomedical image reconstruction. It

will open the door to interact with engineers and researchers and improve my knowledge of GPU technologies, therefore significantly improving the outcomes of my work. I also believe that my research would provide insights for other people to explore the many CUDA-enabled applications in the biomedical scientific domain. Finally, the proposed directions are expected to help scientists and other related fields' researchers to grasp their complex data through interactive computation and visualization.

10.2 Limitations and Cautions of the current work

Despite the fact that the current work has potential impact in the field of bio-medical imaging, it also poses some limitations and cautions when one attempts to use it as a black box without diving deep inside to the methodologies. As can be seen in the conventional methods using iterative scheme, it introduced a computational overhead either from enhancing the **measurement** or restoring the **insight** and therefore leads to quality trade-offs. Even though training adversarially with a cyclic deep neural network can generate much more realistic results, it can also output the non-physical image in the case that the training set is bias to one domain (normal MR images) versus the unseen domain (abnormal MR images that indicate cancer). One promising direction is to give the ability for each perceptron (locally) or the entire neural network (globally) can output the results with some uncertainties for its decisions. In addition, it is really easy to fool the deep neural network model by embedding the gradients of the counterfeit image to the target. Therefore, this kind of attack must be accompanied by a defense mechanism.

On the other hands, although the current work leveraged the state-of-the-art approaches to solve bio-medical image reconstruction, devising such these methods still incorporates some parameter tuning and architecture choices (e.g., number of feature maps, number of neurons, or activation type at each layer, etc.). Furthermore, the myriads of moving objectives to choose correct loss functions and the initialization should be taken into account. This work, more or less, still targeted to supervised (deep neural network) and unsupervised (dictionary learning) techniques. However, the big picture of true artificial intelligence covers a larger scope with many other areas, such as Bayesian learning, Gaussian processes, etc. Thus, combining these to create a uniquely golden method is still an open-end direction. Last but not least, the (deeper) neural network models are not essentially better than other methods (TV regularizers, dictionary learning, etc.,) when we know well the insights that we are going to reconstruct. The future of harnessing powers of artificial intelligence has much more room to explore, rather than focusing on the artificial hand-crafting feature design. It is hence getting much excitement but introducing many more challenges.

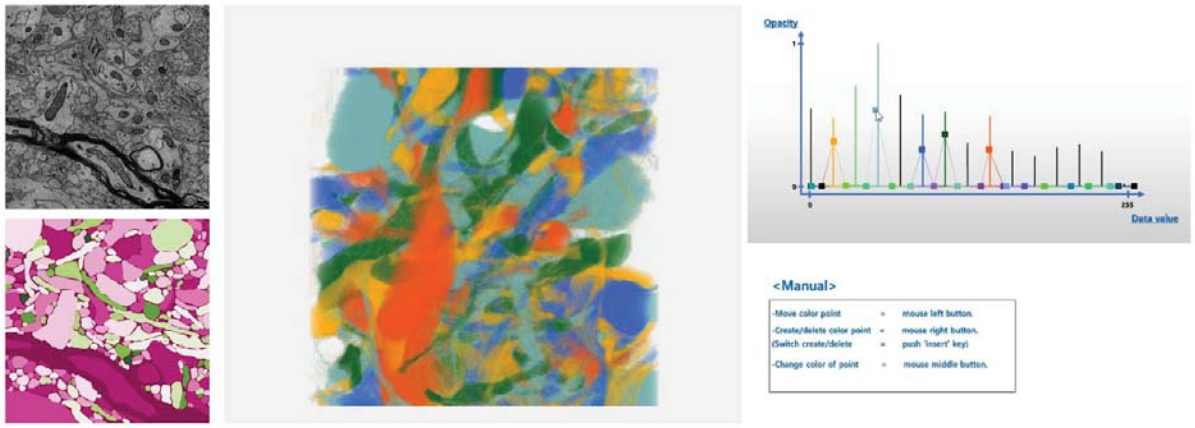


Figure 72: Segmentation module in connectomic segmentation pipeline.

10.3 Future work

Reinforcement Learning (RL) is the area of applying Machine Learning techniques to optimal control problems. RL models usually consist of agents and interactive environments. They focus on acting and decision making at each time step to change the state of the environment in order to achieve an optimal result when the interactivity (episode) finished. The applications of RL are massive-fold, in various domains which need decision making, from robotics to social advertisements. There are interesting examples that can be listed, such as self-learning robotics (Google AI), DeepRL for autonomous driving car, Google HVAC, Recommender systems, Visual question answering (VQA), DeepRL for chatbots, AlphaGo Zero, autoML, JPMorgan, etc. However, gamifying the image processing applications is still not straightforward since the pool of actions is not well defined and its solution is not yet robust compared to end-to-end solutions (e.g., using Fully Convolutional Networks). In the future research work, I wish to use an RL algorithm (Deep Q Network or more advanced methods) to solve an image segmentation problem for connectomic analysis pipeline, which is still in an early stage of interest, by working to design a proper set of actions.

In connectomics research, seminal work has focused on the reconstruction of neuronal wiring diagrams from 3D electron microscopy images of neural tissue. The brain is imaged at high resolution with an electron microscope, and then several image analysis problems need to be solved in order to generate a wiring diagram of the portion of the brain that was imaged: aligning consecutive 2D image slices into a coherent 3D volume, tracing wires to their parent cell body, detecting and characterizing synapses. Among them, the segmentation part (Figure 72) is the most time-consuming since it required domain expertise to complete, even for a small portion of the brain.

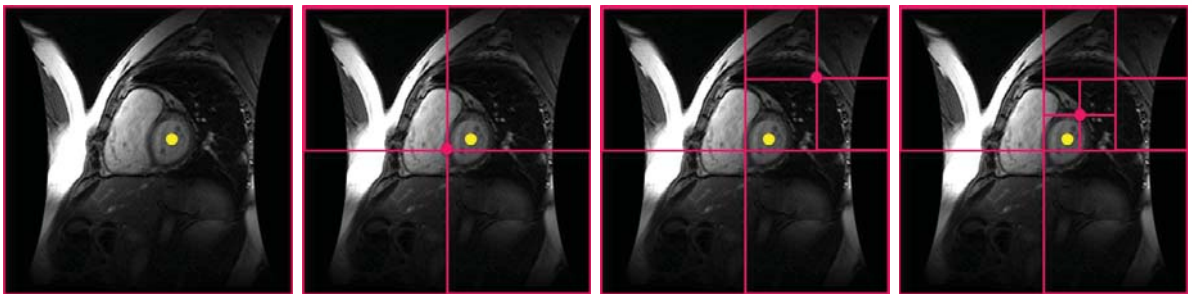


Figure 73: Quadtree decomposition for an agent can solve a landmark detection problem

And there are several solutions that can perform this task well but they are lacking of the ability of correcting the segmentation at different scales. Therefore, developing such an environment and leverage freelancers to join in this platform to annotate each individual neuronal structure is required [22]). Nevertheless, we need to pass a certain levels of experience(s) by achieving positive results compared to neuroscience experts' manual annotations before actually being accepted to perform the segmentation on other regions of the brain. Perhaps, the most recent method that implicitly tackles the connectomic segmentation problem automatically is Flood-filling Network published by Google [68], in which an agent attempts to refine and completely segment a single neuronal structure overtime before starting to delineate other instances.

To begin addressing this proposal to solve the connectomic segmentation problem in an interactive gaming environment, I have implemented a simple demo example to find the center of heart (landmark detection). This scenario is well-defined in [2], where the authors have built an agent and used Deep Q Network to reduce Euclidean distance overtime when it attempts to navigate pixel-by-pixel within the environment. This leads to the total cost may get into $O(n)$ where n is the number of total pixels in the environment. My solution, on the other hand, is much more efficient than the former when I leverage the Quad-Q-Learning [32] (applied for fractal image compression) to reduce the number of steps that the agent needs to perform, i.e., $O(logn)$. I started to construct a quadtree to represent the image environment and step forward with a random action of choosing the current leaf. The episode will be split further until it reaches the smallest areas (Figure 73). The delayed reward will be evaluated based on the inverse of smaller distance overtime. As results, it outperformed the solution from [2] in term of running times.

I also found that there are similar attributes sharing between landmark localization and a simple heart segmentation problem such as the action pool is still relatively small and finite, i.e., if the current area if this leaf has non-unique mask then the agent need to split and push

[22] `https://eyewire.org/explore`

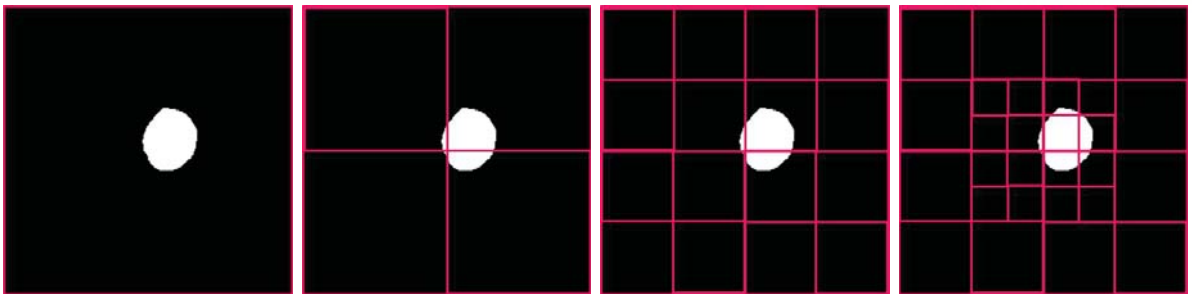


Figure 74: Quadtree decomposition for an agent can solve a heart segmentation problem

those active leaves to the queue, otherwise, it pops the node from the queue and assign either foreground or background (Figure 74).

However, due to my early study on RL and limited amount of time for implementation, the results I got from this demo of heart segmentation, before diving into connectomics data, is not good enough. This can be explained due to the fact the RL excels at decision making but not at pixel assignment in fine-detailed resolution and my current choices of reward is not properly designed. Furthermore, the idea of replacing the network used to approximate the Q-table by Capsule Network [108] is also of great interest.

To end, I wish to continue and pursue this direction, right after my PhD study, in which that I can collaborate with other researchers to accomplish the gamification of connectomics segmentation. I believe that the outcome will make a great contribution not only in the computer vision/machine learning community, but also in neuroscience study.

References

[1] Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322.

[2] Alansary, A., Oktay, O., Yuanwei, L., Le Folgoc, L., Hou, B., Vaillant, G., Glocker, B., Kainz, B., and Rueckert, D. (2018). Evaluating Reinforcement Learning Agents for Anatomical Landmark Detection.

[3] Arganda-Carreras, I., Turaga, S. C., Berger, D. R., Cireşan, D., Giusti, A., Gambardella, L. M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J. M., Liu, T., Seyedhosseini, M., Tasdizen, T., Kamentsky, L., Burget, R., Uher, V., Tan, X., Sun, C., Pham, T. D., Bas, E., Uzunbas, M. G., Cardona, A., Schindelin, J., and Seung, H. S. (2015). Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, page 142.

[4] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.

[5] Beier, T., Andres, B., Köthe, U., and Hamprecht, F. A. (2016). An Efficient Fusion Move Algorithm for the Minimum Cost Lifted Multicut Problem. In *Proceeding of ECCV*, pages 715–730.

[6] Beier, T., Pape, C., Rahaman, N., Prange, T., Berg, S., Bock, D. D., Cardona, A., Knott, G. W., Plaza, S. M., Scheffer, L. K., Koethe, U., Kreshuk, A., and Hamprecht, F. A. (2017). Multicut brings automated neurite segmentation closer to human performance. *Nature Methods*, 14(2):101–102.

[7] Bernabe, G., Cuenca, J., Garcia, L. P., and Gimenez, D. (2014). Improving an autotuning engine for 3d fast wavelet transform on manycore systems. *The Journal of Supercomputing*, 70(2):830–844.

[8] Bhadauria, H. and Dewal, M. (2013). Medical image denoising using adaptive fusion of curvelet transform and total variation. *Computers & Electrical Engineering*, 39(5):1451–1460.

[9] Bilen, C., Wang, Y., and Selesnick, I. (2012). High-speed compressed sensing reconstruction in dynamic parallel MRI using augmented lagrangian and parallel processing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(3):370–379.

[10] Blaimer, M., Breuer, F., Mueller, M., Heidemann, R. M., Griswold, M. A., and Jakob, P. M. (2004). SMASH, SENSE, PILS, GRAPPA: how to choose the optimal method. *Topics in magnetic resonance imaging : TMRI*, 15(4):223–236.

[11] Bock, D. D., Lee, W.-C. A., Kerlin, A. M., Andermann, M. L., Hood, G., Wetzel, A. W., Yurgenson, S., Soucy, E. R., Kim, H. S., and Reid, R. C. (2011). Network anatomy and in vivo physiology of visual cortical neurons. *Nature*, 471(7337):177–182.

[12] Boyd, C. S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*.

[13] Bradley, J. N., Brislawn, C. M., and Hopper, T. (1993). FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression. volume 1961, pages 293–304.

[14] Briggman, K. L., Helmstaedter, M., and Denk, W. (2011). Wiring specificity in the direction-selectivity circuit of the retina. *Nature*, 471(7337):183–188.

[15] Bristow, H., Eriksson, A., and Lucey, S. (2013). Fast Convolutional Sparse Coding. In *Proceeding of IEEE CVPR*, pages 391–398.

[16] Buades, A., Coll, B., and Morel, J. (2005). A non-local algorithm for image denoising. In *Proceedings of CVPR*, volume 2, pages 60–65.

[17] Caballero, J., Price, A., Rueckert, D., and Hajnal, J. (2014a). Dictionary Learning and Time Sparsity for Dynamic MR Data Reconstruction. *IEEE Transactions on Medical Imaging*, 33(4):979–994.

[18] Caballero, J., Price, A. N., Rueckert, D., and Hajnal, J. V. (2014b). Dictionary learning and time sparsity for dynamic MR data reconstruction. *IEEE Transactions on Medical Imaging*, 33(4):979–994.

[19] Caballero, J., Rueckert, D., and Hajnal, J. V. (2012a). Dictionary learning and time sparsity in dynamic MRI. In *Proceeding of MICCAI*, pages 256–263.

[20] Caballero, J., Rueckert, D., and Hajnal, J. V. (2012b). Dictionary Learning and Time Sparsity in Dynamic MRI. In *Proceeding of MICCAI*, number 7510, pages 256–263.

[21] Candes, E. and Wakin, M. (2008). An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30.

[22] Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., Tomancak, P., and Hartenstein, V. (2010). An Integrated Micro- and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy. *PLOS Biol*, 8(10).

[23] Cardona, A., Saalfeld, S., Schindelin, J., Arganda-Carreras, I., Preibisch, S., Longair, M., Tomancak, P., Hartenstein, V., and Douglas, R. J. (2012). TrakEM2 Software for Neural Circuit Reconstruction. *PLOS ONE*, 7(6):e38011.

[24] Carrera, D., Boracchi, G., Foi, A., and Wohlberg, B. (2017). Sparse Overcomplete Denoising: Aggregation Versus Global Optimization. *IEEE Signal Processing Letters*, 24(10):1468–1472.

[25] Chaâri, L., Pesquet, J.-C., Benazza-Benyahia, A., and Ciuciu, P. (2008). Autocalibrated regularized parallel MRI reconstruction in the wavelet domain. In *Proceeding of ISBI*, pages 756–759.

[26] Chambolle, A. (2004). An Algorithm for Total Variation Minimization and Applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97.

[27] Chen, C., Li, Y., and Huang, J. (2013). Calibrationless parallel MRI with joint total variation regularization. In *Proceeding of MICCAI*, pages 106–114.

[28] Chen, H., Qi, X., Cheng, J., and Heng, P. (2016). Deep contextual networks for neuronal structure segmentation. In *Proceeding of AAAI*, pages 1167–1173.

[29] Chen, J., Yang, L., Zhang, Y., Alber, M., and Chen, D. Z. Combining Fully Convolutional and Recurrent Neural Networks for 3d Biomedical Image Segmentation. In *Proceedings of NIPS 2016*, pages 3036–3044.

[30] Cicek, O., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Proceeding of MICCAI*, pages 424–432.

[31] Ciresan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In *Proceeding of NIPS*, pages 2843–2851.

[32] Clausen, C. and Wechsler, H. (2000). Quad-q-learning. *IEEE Transactions on Neural Networks*, 11(2):279–294.

[33] Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

[34] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095.

[35] Daubechies, I. (1992). *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics.

[36] Desmouliers, C., Oruklu, E., and Saniie, J. (2011). Discrete wavelet transform realisation using run-time reconfiguration of field programmable gate array (FPGA)s. *IET Circuits, Devices Systems*, 5(4):321–328.

[37] Donoho, D. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306.

[38] Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., and Pal, C. (2016). The Importance of Skip Connections in Biomedical Image Segmentation. In *Proceedings of DLMIA 2016*, pages 179–187.

[39] Elad, M. and Aharon, M. (2006). Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745.

[40] Enfedaque, P., Auli-Llinas, F., and Moure, J. (2015). Implementation of the DWT in a GPU through a register-based strategy. *IEEE Transactions on Parallel and Distributed Systems*, 26(12):3394–3406.

[41] Ensafi, S., Lu, S., Kassim, A. A., and Tan, C. L. (2014). 3D reconstruction of neurons in electron microscopy images. In *Proceeding of IEEE EMBS*, pages 6732–6735.

[42] Fakhry, A., Peng, H., and Ji, S. (2016). Deep models for brain EM image segmentation: novel insights and improved performance. *Bioinformatics*, 32(15):2352–2358.

[43] Fakhry, A., Zeng, T., and Ji, S. (2017). Residual Deconvolutional Networks for Brain Electron Microscopy Image Segmentation. *IEEE Transactions on Medical Imaging*, 36(2):447–456.

[44] Fatehi, E. and Gratz, P. (2014). ILP and TLP in Shared Memory Applications: A Limit Study. In *Proceeding of the 23rd International Conference on Parallel Architectures and Compilation*, PACT '14, pages 113–126, New York, NY, USA. ACM.

[45] Franco, J., Bernabe, G., Fernandez, J., and Acacio, M. (2009). A parallel implementation of the 2D wavelet transform using CUDA. In *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 111–118.

[46] Franco, J., Bernabé, G., Fernández, J., and Ujaldón, M. (2010). Parallel 3D fast wavelet transform on manycore GPUs and multicore CPUs. *Procedia Computer Science*, 1(1):1101–1110.

[47] Gai, J., Obeid, N., Holtrop, J. L., Wu, X.-L., Lam, F., Fu, M., Haldar, J. P., Hwu, W.-m. W., Liang, Z.-P., and Sutton, B. P. (2013). More IMPATIENT: a gridding-accelerated toeplitz-based strategy for non-cartesian high-resolution 3D MRI on GPUs. *Journal of parallel and distributed computing*, 73(5):686–697.

[48] Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M., and Rossi, F. (2003). *GNU Scientific Library: Reference Manual*. Network Theory Ltd.

[49] Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. In *Proceeding of IEEE CVPR*, pages 2414–2423.

[50] Girshick, R. (2015). Fast R-CNN. In *Proceedings of IEEE ICCV*, pages 1440–1448.

[51] Goldstein, T. and Osher, S. (2009). The split bregman method for $\ell 1$–regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343.

[52] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Proceeding of NIPS*, pages 2672–2680.

[53] Graham, B. J., Hildebrand, D. G. C., and Lee, W.-C. A. (2018). Gridtape imaging stage.

[54] Han, Y. and Ye, J. C. (2018). Framing u-net via deep convolutional framelets: Application to sparse-view ct. *IEEE Transactions on Medical Imaging*, 37(6):1418–1429.

[55] Hayworth, K. J., Morgan, J. L., Schalek, R., Berger, D. R., Hildebrand, D. G. C., and Lichtman, J. W. (2014). Imaging ATUM ultrathin section libraries with WaferMapper: a multi-scale approach to EM reconstruction of neural circuits. *Frontiers in Neural Circuits*, 8.

[56] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of IEEE ICCV*, pages 2961–2969.

[57] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceeding of IEEE CVPR*, pages 770–778.

[58] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *Proceedings of IEEE CVPR*, pages 770–778.

[59] Helmstaedter, M. (2013). Cellular-resolution connectomics: challenges of dense neural circuit reconstruction. *Nature methods*, 10(6):501–7.

[60] Hildebrand, D. G. C. (2015). *Whole-brain functional and structural examination in larval zebrafish*. PhD thesis, Harvard University, Graduate School of Arts and Sciences, Cambridge, MA.

[61] Hildebrand, D. G. C., Cicconet, M., Torres, R. M., Choi, W., Quan, T. M., Moon, J., Wetzel, A. W., Scott Champion, A., Graham, B. J., Randlett, O., Plummer, G. S., Portugues, R., Bianco, I. H., Saalfeld, S., Baden, A. D., Lillaney, K., Burns, R., Vogelstein, J. T., Schier, A. F., Lee, W.-C. A., Jeong, W.-K., Lichtman, J. W., and Engert, F. (2017a). Whole-brain serial-section electron microscopy in larval zebrafish. *Nature*, 545(7654):345–349.

[62] Hildebrand, D. G. C., Graham, B. J., and Lee, W.-C. A. (Google Patents, WO2017184621A1, 2017b). Gridtape for fast nanoscale imaging.

[63] Hoffmann, U., Brix, G., Knopp, M. V., Hess, T., and Lorenz, W. J. (1995). Pharmacokinetic mapping of the breast: a new method for dynamic MR mammography. *Magnetic resonance in medicine: official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, 33(4):506–514.

[64] Hsia, C.-H., Chiang, J.-S., and Guo, J.-M. (2013). Memory-efficient hardware architecture of 2-d dual-mode lifting-based discrete wavelet transform. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(4):671–683.

[65] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of IEEE CVPR*, pages 2261–2269.

[66] Isin, A., Direkoglu, C., and Sah, M. (2016). Review of MRI-based brain tumor image segmentation using deep learning methods. *Procedia Computer Science*, 102:317 – 324.

[67] Isola, P., Zhu, J., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceeding of CVPR*, pages 5967–5976.

[68] Januszewski, M., Kornfeld, J., Li, P. H., Pope, A., Blakely, T., Lindsey, L., Maitin-Shepard, J., Tyka, M., Denk, W., and Jain, V. (2018). High-precision automated reconstruction of neurons with flood-filling networks. *Nature Methods*, 15(8):605–610.

[69] Jeong, W.-K., Beyer, J., Hadwiger, M., Blue, R., Law, C., Vázquez-Reina, A., Reid, R. C., Lichtman, J., and Pfister, H. (2010). Ssecrett and NeuroTrace: Interactive Visualization and Analysis Tools for Large-Scale Neuroscience Data Sets. *IEEE Computer Graphics and Applications*, 30(3):58–70.

[70] Jung, H., Sung, K., Nayak, K. S., Kim, E. Y., and Ye, J. C. (2009). k–t FOCUSS: a general compressed sensing framework for high resolution dynamic MRI. *Magnetic Resonance in Medicine*, 61(1):103–116.

[71] Jung, H., Ye, J. C., and Kim, E. Y. (2007). Improved k–t BLAST and k–t SENSE using FOCUSS. *Physics in Medicine and Biology*, 52(11):3201.

[72] Kang, E., Koo, H. J., Yang, D. H., Seo, J. B., and Ye, J. C. (2018). Cycle Consistent Adversarial Denoising Network for Multiphase Coronary CT Angiography. *arXiv preprint arXiv:1806.09748*.

[73] Kasthuri, N., , K. J., Berger, D. R., Schalek, R. L., Conchello, J. A., Knowles-Barley, S., Lee, D., Vázquez-Reina, A., Kaynig, V., Jones, T. R., Roberts, M., Morgan, J. L., Tapia, J. C., Seung, H. S., Roncal, W. G., Vogelstein, J. T., Burns, R., Sussman, D. L., Priebe, C. E., Pfister, H., and Lichtman, J. W. (2015). Saturated Reconstruction of a Volume of Neocortex. *Cell*, 162(3):648–661.

[74] Kim, D., Trzasko, J., Smelyanskiy, M., Haider, C., Dubey, P., and Manduca, A. (2011). High–performance 3D compressive sensing MRI reconstruction using many–core architectures. *Journal of Biomedical Imaging*, 2011:2:1–2:11.

[75] Kim, T., Cha, M., Kim, H., Lee, J., and Kim, J. (2017). Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *arXiv preprint arXiv:1703.05192*.

[76] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

[77] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Proceeding of NIPS*, pages 1097–1105.

[78] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[79] Lee, D., Yoo, J., and Ye, J. C. (2017). Deep residual learning for compressed sensing MRI. In *Proceeding of IEEE ISBI*, pages 15–18.

[80] Lee, K., Zlateski, A., Ashwin, V., and Seung, H. S. Recursive Training of 2d-3d Convolutional Networks for Neuronal Boundary Prediction. In *Proceedings of NIPS 2015*, pages 3573–3581.

[81] Lee, W.-C. A., Bonin, V., Reed, M., Graham, B. J., Hood, G., Glattfelder, K., and Reid, R. C. (2016). Anatomy and function of an excitatory network in the visual cortex. *Nature*, 532(7599):370–374.

[82] Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., and Aila, T. (2018). Noise2Noise: Learning image restoration without clean data. In *Proceedings of ICML*, pages 2965–2974.

[83] Lichtman, J. W. and Denk, W. (2011). The Big and the Small: Challenges of Imaging the Brain's Circuits. *Science*, 334(6056):618–623.

[84] Lin, M., Chen, Q., and Yan, S. Network in network. In *Proceeding of ICLR 2014*.

[85] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceeding of IEEE CVPR*, pages 3431–3440.

[86] Lustig, M., Donoho, D., and Pauly, J. M. (2007). Sparse MRI: the application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195.

[87] Lustig, M., Donoho, D., Santos, J., and Pauly, J. (2008). Compressed sensing MRI. *IEEE Signal Processing Magazine*, 25(2):72–82.

[88] Mallat, S. (2008). *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition.

[89] Mallat, S. and Hwang, W. (1992). Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643.

[90] Manjón, J. V., Carbonell-Caballero, J., Lull, J. J., García-Martí, G., Martí-Bonmatí, L., and Robles, M. (2008). MRI denoising using non-local means. *Medical image analysis*, 12(4):514–523.

[91] Mardani, M., Gong, E., Cheng, J. Y., Vasanawala, S., Zaharchuk, G., Alley, M., Thakur, N., Han, S., Dally, W., Pauly, J. M., et al. (2017). Deep Generative Adversarial Networks for Compressed Sensing Automates MRI. *arXiv preprint arXiv:1706.00051*.

[92] Matela, J. (2009). GPU–Based DWT acceleration for JPEG2000. *Annual Doctoral Workshop On Mathematical and Engineering Methods in Computer Science*, pages 136–143.

[93] Micikevicius, P. (2009). 3D finite difference computation on GPUs using CUDA. In *Proceeding of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-2, page 79–84.

[94] Murphy, M., Alley, M., Demmel, J., Keutzer, K., Vasanawala, S., and Lustig, M. (2012). Fast -SPIRiT compressed sensing parallel imaging MRI: scalable parallel implementation and clinically feasible runtime. *IEEE Transactions on Medical Imaging*, 31(6):1250–1262.

[95] Otazo, R., Candès, E., and Sodickson, D. K. (2015). Low-rank plus sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic components. *Magnetic Resonance in Medicine*, 73(3):1125–1136.

[96] Parag, T., Ciresan, D. C., and Giusti, A. Efficient Classifier Training to Minimize False Merges in Electron Microscopy Segmentation. In *Proceeding of IEEE ICCV 2015*, pages 657–665.

[97] Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.

[98] Pizurica, A., Wink, A. M., Vansteenkiste, E., Philips, W., and Roerdink, B. J. (2006). A review of wavelet denoising in MRI and ultrasound brain imaging. *Current medical imaging reviews*, 2(2):247–260.

[99] Quan, T. M., Han, S., Cho, H., and Jeong, W.-K. (2015). Multi-GPU reconstruction of dynamic compressed sensing MRI. In *Proceeding of MICCAI*, pages 484–492.

[100] Quan, T. M. and Jeong, W.-K. (2014). A fast mixed-band lifting wavelet transform on the GPU. In *IEEE International Conference on Image Processing*, pages 1238–1242.

[101] Quan, T. M. and Jeong, W.-K. (2016a). Compressed sensing dynamic MRI reconstruction using GPU-accelerated 3D convolutional sparse coding. In *Proceeding of MICCAI*, pages 484–492.

[102] Quan, T. M. and Jeong, W.-K. (2016b). Compressed sensing reconstruction of dynamic contrast enhanced MRI using GPU-accelerated convolutional sparse coding. In *Proceeding of IEEE ISBI*, pages 518–521.

[103] Quan, T. M., Nguyen-Duc, T., and Jeong, W.-K. (2018). Compressed Sensing MRI Reconstruction Using a Generative Adversarial Network With a Cyclic Loss. *IEEE Transactions on Medical Imaging*, 37(6):1488–1497.

[104] Ravishankar, S. and Bresler, Y. (2011). MR image reconstruction from highly undersampled k-space data by dictionary learning. *IEEE Transactions on Medical Imaging*, 30(5):1028–1041.

[105] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceeding of MICCAI*, pages 234–241.

[106] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

[107] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

[108] Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules.

[109] Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J.-Y., White, D. J., Hartenstein, V., Eliceiri, K., Tomancak, P., and Cardona, A. (2012). Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9(7):676–682.

[110] Schlemper, J., Caballero, J., Hajnal, J. V., Price, A. N., and Rueckert, D. (2017). A deep cascade of convolutional neural networks for dynamic mr image reconstruction. *IEEE Transactions on Medical Imaging*, 37(2):491–503.

[111] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[112] Skodras, A., Christopoulos, C., and Ebrahimi, T. (2001). The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58.

[113] Sommer, C., Strähle, C., Köthe, U., and Hamprecht, F. A. (2011). ilastik: Interactive learning and segmentation toolkit. In *Proceeding of IEEE ISBI*, pages 230–233. 1.

[114] Song, C., Li, Y., Guo, J., and Lei, J. (2014). Block-based two-dimensional wavelet transform running on graphics processing unit. *IET Computers Digital Techniques*, 8(5):229–236.

[115] Song, C., Li, Y., and Huang, B. (2011). A GPU-accelerated wavelet decompression system with SPIHT and reed-solomon decoding for satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):683–690.

[116] Stollenga, M. F., Byeon, W., Liwicki, M., and Schmidhuber, J. (2015). Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Proceeding of NIPS*, pages 2998–3006.

[117] Sun, J., Li, H., Xu, Z., et al. (2016). Deep ADMM-net for compressive sensing MRI. In *Proceeding of NIPS*, pages 10–18.

[118] Sweldens, W. (1998). The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546.

[119] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceeding of IEEE CVPR*, pages 1–9.

[120] Tian, Z., Jia, X., Yuan, K., Pan, T., and Jiang, S. B. (2011). Low-dose ct reconstruction via edge-preserving total variation regularization. *Physics in medicine and biology*, 56(18):5949.

[121] Tobin, W. F., Wilson, R. I., and Lee, W.-C. A. (2017). Wiring variations that enable and constrain neural computation in a sensory microcircuit. *eLife*, 6.

[122] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Proceedings of ICCV*, pages 839–846.

[123] Trémoulhéac, B., Dikaios, N., Atkinson, D., and Arridge, S. R. (2014). Dynamic MR Image Reconstruction–Separation From Undersampled (k-t)-Space via Low-Rank Plus Sparse Prior. *IEEE Transactions on Medical Imaging*, 33(8):1689–1701.

[124] Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., and Seung, H. S. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Comput.*, 22(2):511–538.

[125] van der Laan, W., Jalba, A., and Roerdink, J. B. T. M. (2011). Accelerating wavelet lifting on graphics hardware using CUDA. *IEEE Transactions on Parallel and Distributed Systems*, 22(1):132–146.

[126] van der Laan, W., Roerdink, J. B. T. M., and Jalba, A. (2009). Accelerating wavelet-based video coding on graphics hardware using CUDA. In *Proceeding of 6th International Symposium on Image and Signal Processing and Analysis, 2009. ISPA 2009*, pages 608–613.

[127] Volkov, V. and Demmel, J. (2008). Benchmarking GPUs to tune dense linear algebra. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–11.

[128] Wang, S., Su, Z., Ying, L., Peng, X., Zhu, S., Liang, F., Feng, D., and Liang, D. (2016). Accelerating magnetic resonance imaging via deep learning. In *Proceeding of IEEE ISBI*, pages 514–517.

[129] Wetzel, A. W., Bakal, J., Dittrich, M., Hildebrand, D. G. C., Morgan, J. L., and Lichtman, J. W. Registering large volume serial-section electron microscopy image sets for neural circuit reconstruction using fft signal whitening. In *Proceedings of AIPR Workshop 2016*.

[130] Wiehman, S. and Villiers, H. d. Semantic segmentation of bioimages using convolutional neural networks. In *Proceeding of IJCNN 2016*, pages 624–631.

[131] Wohlberg, B. (2014). Efficient convolutional sparse coding. In *Proceeding of IEEE ICASSP*, pages 7173–7177.

[132] Wohlberg, B. (2016). Efficient algorithms for convolutional sparse representations. *IEEE Transactions on Image Processing*, 25(1):301–315.

[133] Wu, X. (2015). An iterative convolutional neural network algorithm improves electron microscopy image segmentation. *CoRR*, abs/1506.05849.

[134] Yang, G., Yu, S., Dong, H., Slabaugh, G., Dragotti, P. L., Ye, X., Liu, F., Arridge, S., Keegan, J., Guo, Y., and Firmin, D. (2018). Dagan: Deep de-aliasing generative adversarial networks for fast compressed sensing mri reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1310–1321.

[135] Yao, J., Xu, Z., Huang, X., and Huang, J. (2015). Accelerated dynamic MRI reconstruction with total variation and nuclear norm regularization. In *Proceeding of MICCAI*, pages 635–642.

[136] Zeiler, M. D. and Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In *Proceeding of ECCV*, pages 818–833.

[137] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *Proceeding of IEEE CVPR*, pages 2528–2535.

[138] Zheng, Y., Liu, D., Georgescu, B., Nguyen, H., and Comaniciu, D. (2015). 3D Deep Learning for Efficient and Robust Landmark Detection in Volumetric Data. In *Proceeding of MICCAI*, pages 565–572.

[139] Zheng, Z., Lauritzen, J. S., Perlman, E., Robinson, C. G., Nichols, M., Milkie, D., Torrens, O., Price, J., Fisher, C. B., Sharifi, N., Calle-Schuler, S. A., Kmecova, L., Ali, I. J., Karsh, B., Trautman, E. T., Bogovic, J. A., Hanslovsky, P., Jefferis, G. S. X. E., Kazhdan, M., Khairy, K., Saalfeld, S., Fetter, R. D., and Bock, D. D. (2018). A Complete Electron Microscopy Volume of the Brain of Adult Drosophila melanogaster. *Cell*, 174(3):730–743.e22.

[140] Zhu, J., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceeding of ICCV*, pages 2242–2251.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor Won-Ki Jeong for accepting me as his student. I would have never completed the PhD program and this dissertation without his extensive, constant guidance and support. I own him my deepest thanks for his patience and trust when he allowed me to explore many interesting directions in my research, brought me to diverse places across the globe to work with other smart people as well as expose and share the knowledge.

Many thanks as well to my committee to Professor Jaesik Choi (UNIST), Professor Se Young Chun (UNIST), Professor Hyung Joon Cho (UNIST) and Professor Wei-Chung Allen Lee (Harvard Medical School) for their support, advice and suggestions during the preparation of this dissertation. I would also like to thank the rest of the faculty members and staffs in School of Electrical Computer Engineering for their endless efforts to make UNIST a unique and fun place to study and work.

I would also like to express my appreciation to my co-authors, David Grant Colburn Hildebrand (The Rockefeller University); Sohyun Han (UNIST); Logan A. Thomas, Aaron T. Kuan (Harvard Medical School) and my colleagues, Donglai Wei, Won-Dong Jang (Visual Computing Group, Harvard University); Doan Xuan Quang Minh (Broad Institute of Harvard and MIT), Mai Xuan Toan (Massachusetts General Hospital) for their supports and research collaboration.

In addition, I would like to give thanks to my Labmates (Hyungsuk Choi, Sumin Hong, Woohyuk Choi, Gyuhyun Lee, Junyoung Choi, HaeJin Jeong, Kanggeun Lee, Jungmin Moon, Inhwan Yoo, Nguyen Duc Van Thanh, Hyungjoon Jang, Nguyen Thi Mai Huong, Tran Anh Tuan) and Friends for their generous helps, both in research and in life.

Last but not least, I would like to thank my Family, especially my Parents and Relatives for their encouragement, unconditional love and staying nicely when I have spent all my nights, either weekdays or weekends, for doing this work. And my deepest gratitude to Nguyen Thi Hoang Anh for talking to me, cheering me up when I was struggling around with stresses and desperations.

To the living entities that live under the sun and dream to fly to the moon; also the brightest stars produced by the darkest sky, thanks for being there.